2021-1542

IN THE UNITED STATES COURT OF APPEALS FOR THE FEDERAL CIRCUIT

SAS INSTITUTE INC.,

Plaintiff-Appellant,

v.

WORLD PROGRAMMING LIMITED,

Defendant-Appellee.

Appeal from the United States District Court for the Eastern District of Texas in Case No. 2:18-cv-00295-JRG, Hon. J. Rodney Gilstrap, Chief Judge

BRIEF OF *AMICUS CURIAE* GITHUB, INC. IN SUPPORT OF DEFENDANT-APPELLEE AND AFFIRMANCE

Joseph C. Gratz
Samuel J. Zeitlin
Durie Tangri LLP
217 LEIDESDORFF STREET
SAN FRANCISCO, CA 94111
Telephone: 415-362-6666
Facsimile: 415-236-6300

August 30, 2021

Attorneys for *Amicus Curiae* GitHub, Inc.

CERTIFICATE OF INTEREST

Counsel for *Amicus Curiae* certify as follows:

1. The full name of every party represented by us is:

GitHub, Inc.

2. The name of the real party in interest represented by us is:

Parties Represented	Real Party in Interest
GitHub, Inc.	N/A

3. All parent corporations and publicly held companies that own 10% or more of stock in the parties represented by us are:

Parties Represented	All parent corporations and any public companies that own 10 percent or more of the stock of the party represented by us
GitHub, Inc.	Microsoft Corporation

4. The names of all law firms and the partners or associates that appeared for the party or amicus now represented by me in the trial court or agency or are expected to appear in this Court (and who have not or will not enter an appearance in this case) are:
Durie Tangri LLP: Joseph C. Gratz; and Samuel J. Zeitlin

5. The title and number of any case known to counsel to be pending in this or any other court or agency that will directly affect or be directly affected by this Court's decision in the pending appeal.

N/A

Respectfully submitted,

/s/ Joseph C. Gratz

Joseph C. Gratz
Samuel J. Zeitlin
Durie Tangri LLP
217 LEIDESDORFF STREET
SAN FRANCISCO, CA 94111
Telephone: 415-362-6666
Facsimile: 415-236-6300

August 30, 2021

TABLE OF CONTENTS

Page

TABLE OF AUTHORITIES

Page(s)

Cases

Statutes

Other Authorities

INTEREST OF *AMICUS CURIAE*[1]

GitHub, Inc. runs the world's largest online software development platform, enabling more than 65 million individual developers, students, startups, small businesses, large companies, NGOs, and governments to host and collaborate on software projects. Much of that software is open source: freely available for anyone to use, study, modify, or distribute for any purpose.

GitHub makes it easier for developers to be developers: to work together, to solve challenging problems, to build on one another's work and create the world's most important technologies. Because software code is increasingly developed and distributed on platforms such as GitHub, GitHub's interest here is to support developers (who are users of its software collaboration platform) and their software innovations when claims of copyright infringement are made against their projects, often creating copyright law FUD (fear, uncertainty, and doubt).

Vague allegations of nonliteral copyright infringement especially create FUD because the recipients of such accusations frequently have no way to evaluate the risk they are facing. These sorts of allegations often have a greater impact on software than other artistic works such as photos or books. When software is

[1] All parties have consented to the filing of this brief. *Amicus Curiae* and its counsel certifies that no counsel for either party authored this brief in whole or in part, and that no one other than *Amicus* made a monetary contribution to the brief's preparation or submission.

removed from a collaboration platform like GitHub in response to an infringement complaint, its removal can impact an exponential number of users beyond the developer, because in modern software development, programmers write code that "depends" on other tested, proven, and widely accessible software—usually open source software—written by third parties. All types of software, from the apps on your phone and laptop to enterprise software run by corporations and governments, rely on these "dependencies."

When a heavily depended-on piece of software is taken offline due to vague allegations of copyright infringement, the entire ecosystem relying on that piece breaks. GitHub asks the Court to uphold the trial court's approach to require nonliteral software copyright plaintiffs to clearly articulate exactly what they are claiming was copied at early, practicable stages. This approach protects software developers as well as the innovation ecosystem they build and the world depends on. This approach also appropriately protects copyright owner interests by encouraging legitimate infringements to be quickly corrected or removed, while helping to prevent non-infringing code from being impacted.

INTRODUCTION AND STATEMENT OF ARGUMENT

Imagine that you are a software developer. You receive a demand letter from a rival firm, claiming that your product infringes its copyrighted software. Naturally, you respond by asking what part of their software you have allegedly infringed, and how. "Everything about my software is creative," says your rival. "You infringe something, but I won't tell you what until trial."

Next, imagine you are a third-party collaboration platform hosting code that is alleged to infringe copyrighted software. You receive an infringement notice demanding takedown of code hosted on your platform—code that has likely been copied into hundreds or thousands of other repositories on your platform and is used in the myriad of projects also hosted there. To preserve your immunity from liability under copyright law, you must act expeditiously either to remove access to the claimed infringing material, or require the developer quickly to remediate and repost the remediated code so that it is no longer infringing. You ask the claimant to specify or provide representative examples of infringement and you receive a similar non-response. You are faced with two bad choices: if you take no action, you risk losing your safe harbor as to that code, but if you remove code that is non-infringing, you may cause significant disruption for developers across your platform.

This is a recipe for mischief. Litigants cannot be permitted to shepherd

claims that are deficient as a matter of law all the way to trial under cover of obscurity. And courts need not permit them to do so. Copyright infringement claims brought over the structure, sequence, and organization of software (nonliteral infringement claims) frequently raise the problem of nebulous, shifting theories of infringement. But this is a problem that intellectual property law has repeatedly confronted. The solution—in copyright, trade secret, and patent law—has been to require plaintiffs to disclose their infringement theories and representative examples with particularity, as early as is practicable. In the litigation context, Judge Gilstrap appropriately exercised his discretion to require SAS to identify what material SAS owned that WPL allegedly infringed.

Early disclosure of theories of nonliteral software copyright infringement is of critical importance to GitHub and its users in the open source and broader software development communities. Software development is a collaborative process. Wherever possible, the best practice is to rely on tested, proven open source code created and made publicly available on platforms like GitHub. *All* developers—from hobbyists to startups to "big tech"—depend on small libraries maintained by third parties under an open source license. When small developers are threatened with vague nonliteral copyright infringement allegations, the impact is felt across the whole developer ecosystem.

Requiring claimants in nonliteral software infringement cases to identify, as

early as practicable, *what protectible material was copied* encourages efficient dispute-resolution between parties outside of litigation. When provided with knowledge about what portions allegedly infringe, developers can choose to remove allegedly infringing code and ensure non-infringing code remains available without severely impacting their users and the broader software ecosystem.

That same approach provides roadmaps to platforms like GitHub when they regularly inspect infringement notices to ensure that they comply with requirements of section 512 of the Digital Millennium Copyright Act (DMCA), requirements that ensure developers have sufficient information to be able to respond effectively. Code on GitHub may be in use by millions of computers around the world, and a wrongful or malicious takedown can have enormous consequences to GitHub's users in the developer ecosystem. Accordingly, copyright claimants and developers frequently look to case law to understand the legal requirements and burdens necessary to demonstrate infringement, especially when evaluating notices claiming nonliteral infringement. Case law also helps guide platforms such as GitHub in determining whether a copyright infringement notice is legally compliant with copyright law such that the code should be removed based solely on the allegations in that copyright infringement notice, and to ensure that its users have, where possible, an opportunity expeditiously to fix their code before GitHub takes it down.

ARGUMENT

I. Permitting plaintiffs to proceed with conclusory assertions of nonliteral infringement facilitates copyright FUD and disruption of software development

Judge Gilstrap held a copyrightability hearing in this case because he wanted to ensure that the Court, the parties, and the jury would have clarity about what protectable expression in SAS's program SAS believes WPL infringed. *See* Appx1-2. He dismissed the claim after the hearing because he found that SAS's expert refused to actually identify any such expression; all the expert did was repeat over and over that everything about SAS's software was "creative." Appx17.

Early and particularized disclosure of infringement theories creates clarity and prevents mischief in the world of software development. GitHub's thriving community of 65 million open source software developers includes amateurs and individuals all the way up to the largest tech and Fortune 500 companies, with code spread across 200 million repositories. Because modern software development is interdependent, with shared functionality dispersed among these many projects, a vague claim against one project can cause widespread disruption. Moreover, permitting vague nonliteral software copyright infringement claims to proceed to trial would hinder GitHub's own efforts to allow its users fairly and expeditiously to resolve copyright disputes on its platform at much earlier stages by encouraging

copyright claimants to provide as little information as possible.

A. Allowing vague infringement allegations to proceed encourages copyright claimants to send vague infringement notices, which can cause disruption to the software community more broadly.

GitHub is the home of the open source development community online. Open source projects are free to host and maintain on GitHub, and there are hundreds of millions of such projects on the platform. Developers use platforms like GitHub to distribute their code to millions of users, and GitHub's role in hosting code online—particularly open source code maintained by volunteers—means that software copyright disputes often begin and end when copyright owners send infringement notices to GitHub demanding that the allegedly infringing code be removed.

GitHub takes its responsibility to the developer community seriously. It examines every notice of infringement it receives to ensure that 17 U.S.C. § 512(c)(3)(A)(iii)'s requirement is met: that the notice specifies the allegedly infringing content so that it can be identified. In most cases, the only information about the alleged infringement is the information contained in the notice, and if that information proves insufficient to locate the infringing material, GitHub may ask the copyright claimant to provide more information. Identifying infringing content with sufficient specificity ensures that developers have fair notice and are not faced with vague, overbroad, mistaken, or abusive takedown notices. These

steps also limit the likelihood that otherwise valid open source projects are unfairly taken down.

Where an infringement notice alleges that some portion of a software project infringes copyright, GitHub does not immediately shut down the entire repository of code in response. GitHub, DMCA Takedown Policy, https://docs.github.com/en/github/site-policy/dmca-takedown-policy (last visited Aug. 15, 2021). Instead, GitHub contacts the user who created the repository and provides an opportunity to remove or modify the specific source files or content specified in the notice. *Id.* If the developer makes changes expeditiously and notifies GitHub that the alleged infringing material has been addressed, GitHub will then inform the copyright holder, who then has an opportunity to revise or maintain the notice if they are not satisfied with the changes. *Id.*

This process facilitates the speedy removal of allegedly infringing material without causing disruption to the potentially millions of users who depend on widely used code. GitHub's approach depends on receiving notices that identify the nature of the alleged infringement with some degree of specificity. If nonliteral software copyright infringement claims can be taken *all the way to trial* without ever clearly articulating a theory of infringement, there would be little reason for any copyright claimant ever to provide any specificity in any infringement notice in any context.

Allowing copyright claimants to avoid providing this specificity at early practicable stages would discourage efficient approaches like GitHub's of addressing copyright owners' legitimate concerns while limiting the impact on non-infringing materials. Worse, it would incentivize platforms to delete code to maintain their safe harbor in the face of even the vaguest of claims, instead of providing a process that allows developers to quickly address allegations and stand up for their rights.

In most cases, removal of code by a cloud platform would effectively end legitimate projects. From January 2020 to June 2021, GitHub processed 3,078 copyright-related claims, which resulted in the removal of thousands of software projects. These removals were based primarily on the information provided by copyright claimants. Permitting vague allegations of nonliteral copyright infringement to survive summary judgment would incentivize copyright claimants to provide less information about their claim at every stage, starting with initial demand letters and DMCA takedown notices. The result will be that more code is removed without any legal determination, to be restored only in those rare cases that make it through trial. This would create significant opportunities for unscrupulous or simply overzealous copyright holders to disrupt the millions of developers across the open source software ecosystem, based merely on vague allegations of nonliteral infringement.

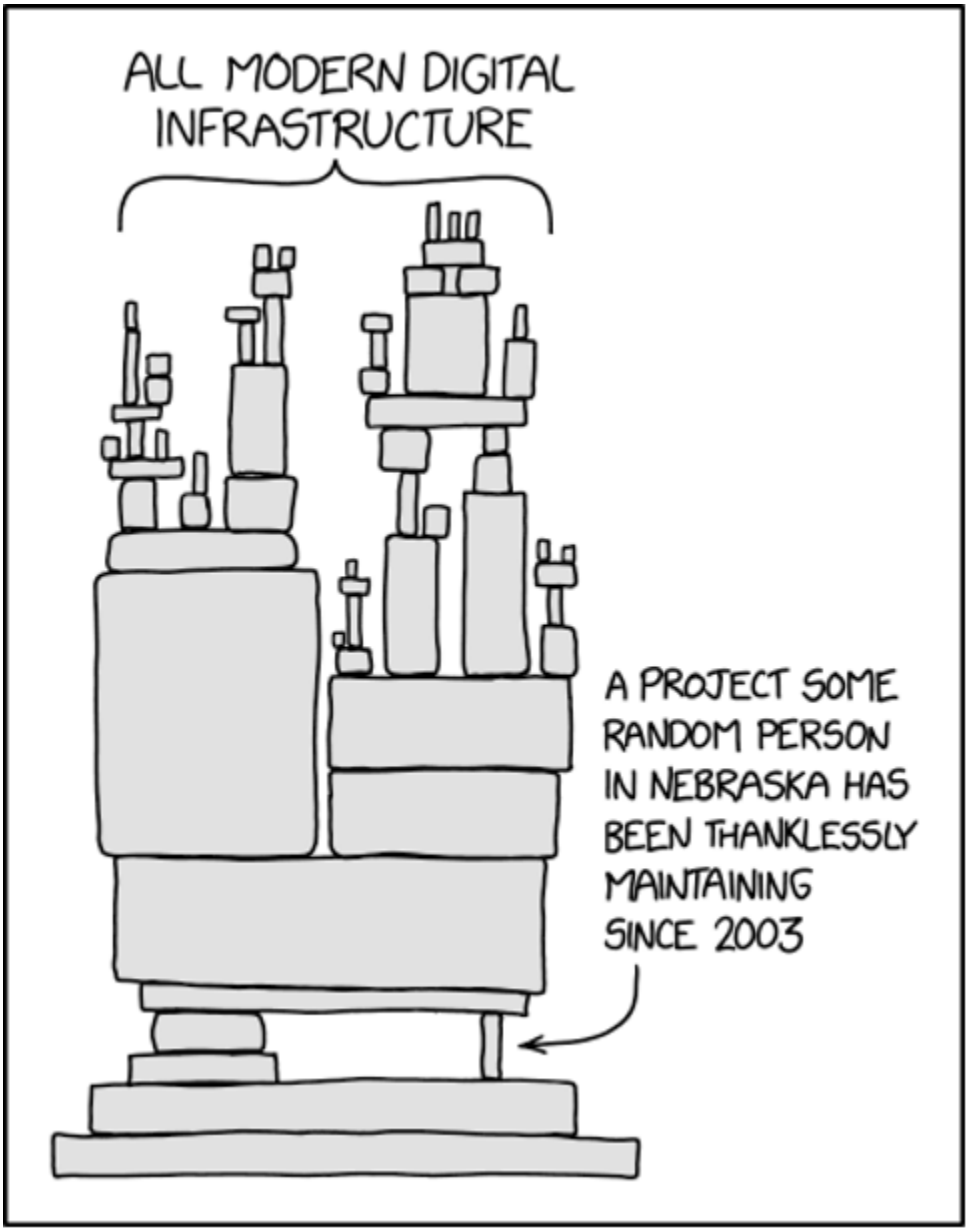
B. Vague infringement allegations create disproportionate risks in the open source software development ecosystem

The open source software community is particularly vulnerable to wrongful or unfounded claims of copyright infringement. Open source software is ubiquitous, relied on in countless contexts by a wide variety of projects and users. As a result, improper copyright takedowns of open source code can have serious and far-reaching consequences.

Open source software is code made available under an open source license for anyone to use, study, modify, and distribute for any purpose. GitHub, Starting an Open Source Project, https://opensource.guide/starting-a-project/ (last visited Aug. 15, 2021). Open source software is powerful because it enables decentralized collaboration between developers around the world to solve problems, build features, and catch bugs, and it has become an essential aspect of modern computing. *Id.* Many of the largest technology companies in the world rely on open source software, and the communities that grow up around open source projects. *See, e.g.*, Microsoft Corp., Open Source, https://opensource.microsoft.com/ (last visited Aug. 15, 2021); Google LLC, Google Open Source, https://opensource.google/ (last visited Aug. 15, 2021); Amazon Web Services, Inc., Open Source at AWS, https://aws.amazon.com/opensource/ (last visited Aug. 15, 2021); Apple Inc., Apple Open Source, https://opensource.apple.com/ (last visited Aug. 15, 2021).

If code on which lots of other software depends is the subject of a wrongful takedown, the consequences for the software ecosystem can be abrupt and disastrous. *See, e.g.*, Keith Collins, *How one programmer broke the internet by deleting a tiny piece of code*, Quartz (March 27, 2016), https://qz.com/646467/how-one-programmer-broke-the-internet-by-deleting-a-tiny-piece-of-code/ (describing how an individual developer's removal of a few lines of code from GitHub in response to a legal demand broke widely used software packages); Thomas Claburn, *Ruby off the Rails: Code library yanked over license blunder, sparks chaos for half a million projects*, The Register (Mar. 25, 2021), https://www.theregister.com/2021/03/25/ruby_rails_code/ (describing how a licensing-related takedown of a widely used software library broke the Ruby on Rails software development framework, along with more than 500,000 other projects).

The "domino effect" of takedowns of code is felt most strongly in projects of open source participants, large and small, which do not do not exist in isolation, but are tied together through a complex web of dependencies—incorporation by reference of code from other open source projects. Even software used in major commercial applications typically depends on code maintained by lone individuals. Below is a visual depiction of this state of affairs by award-winning cartoonist Randall Munroe:



Randall Munroe, *Dependency*, xkcd, https://xkcd.com/2347/ (last visited Aug. 18, 2021) (licensed under CC BY-NC 2.5).

Although some participants in the open source ecosystem are powerful companies well equipped with copyright experts to deal with legally deficient and vague claims, the vast majority of open source developers do not have well-staffed

legal departments at the ready. "Copyright infringement . . . lends itself readily to abusive litigation, since the high cost of trying such a case can force a defendant who might otherwise be successful in trial to settle in order to avoid the time and expenditure of a resource intensive case." *Nat'l Bus. Dev. Servs., Inc. v. Am. Credit Educ. & Consulting Inc.*, 299 F. App'x 509, 512 (6th Cir. 2008).

This is particularly true of nonliteral software copyright claims, which can raise acute problems of fair notice. *See* Part II.A *infra*. Vague accusations of nonliteral software copyright infringement create FUD (fear, uncertainty, and doubt) because developers who receive such claims have no way to analyze them and determine their merit, or quickly remediate their code so that it won't be taken down and disrupt other projects. If such claims can go all the way to trial without specifying exactly what protected material was allegedly copied, it heightens the risk that a wrongful or erroneous copyright claim will disrupt the software development ecosystem in disproportionate ways.

II. Judge Gilstrap appropriately required SAS to articulate a viable theory of copyright infringement prior to trial

The purpose of Judge Gilstrap's "novel copyrightability hearing," Appellant's Br. at 31, was to force SAS to identify at least *some* protectable material in its software that it claimed was taken by WPL. In traditional copyright infringement claims over prose or images, the basis for the claim is usually obvious from a visual comparison of the two works. Nonliteral software copyright

infringement claims are different: by their nature they turn on abstract comparisons of technical subject matter. Without proper judicial supervision, such claims can remain vague and unsubstantiated all the way to trial. Judge Gilstrap rightly exercised his discretion to make sure SAS had a viable theory of copyright infringement before letting SAS bring its case before a jury.

A. Intellectual property law relies on procedural safeguards to protect defendants from vague and shifting theories of infringement like those present in nonliteral software copyright cases

In most copyright cases, the question of *what* the defendant allegedly copied from the plaintiff's work does not generally raise fair notice concerns, as visual comparisons can quickly reveal potential issues. However, nonliteral software copyright infringement cases typically concern abstract similarities in voluminous source code that are not readily understandable to a lay observer, and not easily identifiable even to a software developer without more specific detail.

Recognizing the need for a nonliteral copyright infringement plaintiff to articulate infringement with particularity at early practicable stages, the Fifth Circuit requires a showing that "the allegedly infringing work is substantially similar to protectable elements of the infringed work," using the "abstraction-filtration-comparison" test from the Second Circuit. *Gen. Universal Sys., Inc. v. Lee*, 379 F.3d 131, 142 (5th Cir. 2004) (citing *Comput. Assocs. Int'l, Inc. v. Altai, Inc.*, 982 F.2d 693 (2d Cir. 1992)).

Though portrayed by SAS as "novel", approaches like that used by Judge Gilstrap are not unique to copyright claims. Similar problems arise whenever intellectual property law (whether copyright, trade secret, or patent) deals with complex, technical subject matter and abstract forms of infringement. As a result, the law provides safeguards that require plaintiffs to identify with particularity what they claim to be infringed and infringing at early, practicable stages.

Some of these safeguards are explicitly built into statutes. For example, the notice and takedown regime of the DMCA requires rightsholders to provide "[i]dentification of the material that is claimed to be infringing or to be the subject of infringing activity and that is to be removed or access to which is to be disabled, *and information reasonably sufficient* to permit the service provider to locate the material." 17 U.S.C. § 512(c)(3)(A)(iii) (emphasis added). As described in Part I.B *supra*, GitHub relies on this information in order quickly to resolve copyright disputes with minimal disruption to software that may be relied on by large numbers of users.

In other areas of intellectual property law, safeguards have been established by the courts. In trade secret law, for instance, there is a "general requirement that a person claiming rights in a trade secret bears the burden of defining the information for which protection is sought with sufficient definiteness to permit a court to apply the criteria for protection and to determine the fact of an

appropriation." *Olaplex, Inc. v. L'Oréal USA, Inc.*, No. 2020-1382, 2021 WL 1811722, at *9 (Fed. Cir. May 6, 2021) (quoting *TLS Mgmt. & Mktg. Servs., LLC v. Rodríguez-Toledo*, 966 F.3d 46, 52 (1st Cir. 2020)) (cleaned up). This requirement was necessary "because, without particularity (pre-trial and at trial), there is an inadequate basis for a fair adjudication of what information was actually used by the defendants." *Id.*; *see also InteliClear, LLC v. ETC Glob. Holdings, Inc.*, 978 F.3d 653, 658 (9th Cir. 2020) ("Identifying trade secrets with sufficient particularity is important because defendants need concrete identification to prepare a rebuttal. Courts and juries also require precision because, especially where a trade secrets claim involves a sophisticated and highly complex system, the district court or trier of fact will not have the requisite expertise to define what the plaintiff leaves abstract.") (internal quotation marks and citations omitted); *Givaudan Fragrances Corp. v. Krivda*, 639 F. App'x 840, 845 (3d Cir. 2016) ("It is patently obvious that trade secrets must be identified with enough specificity to put a defendant on notice of what is actually alleged to have been stolen."); *IDX Sys. Corp. v. Epic Sys. Corp.*, 285 F.3d 581, 583–84 (7th Cir. 2002) (plaintiff's "43–page description of the methods and processes underlying and the inter-relationships among various features making up [its] software package" was not sufficiently particular to survive summary judgment because "although the document was created for this litigation, it does not separate the trade secrets from

the other information that goes into any software package."). In many jurisdictions, this identification must be made before even commencing discovery. *See, e.g.*, *Olaplex*, 2021 WL 1811722, at *9 (Delaware law); *Meggitt San Juan Capistrano, Inc. v. Yongzhong*, 575 F. App'x 801, 803 (9th Cir. 2014) (California law).

In patent law, many courts have issued local rules requiring early submission of infringement and invalidity contentions. *See* Xuan-Thao Nguyen, *Dynamic Federalism and Patent Law Reform*, 85 Ind. L.J. 449, 477 (2010). The purpose of these rules, whose application this Court has repeatedly upheld, is "to require parties to crystallize their theories of the case early in the litigation so as to prevent the 'shifting sands' approach to claim construction." *Keranos, LLC v. Silicon Storage Tech., Inc.*, 797 F.3d 1025, 1035 (Fed. Cir. 2015) (internal quotation marks omitted and citation omitted). These rules "further the goal of full, timely discovery and provide all parties with adequate notice of and information with which to litigate their cases." *AntiCancer, Inc. v. Pfizer, Inc.*, 769 F.3d 1323, 1331 (Fed. Cir. 2014) (citation omitted).

The concerns that animate particularity requirements in other areas of intellectual property law apply equally to nonliteral software copyright claims. As in trade secret law, defendants need specificity to rebut plaintiffs' claims, and courts and juries lack the expertise to flesh out what plaintiffs leave vague. And

just as in patent law, nonliteral software copyright infringement litigation creates the opportunity for unfair "shifting sands" theories regarding what abstract elements of the plaintiff's program the defendant allegedly copied.

B. Judge Gilstrap was within his discretion when he required SAS to disclose its theory of nonliteral infringement at a "copyrightability hearing"

Judge Gilstrap ordered the copyrightability hearing in this case to enable "the jury to make a clear and reliable determination of whether infringement exists as to the *asserted non-literal elements* of the computer software at issue . . . ." Appx1-2 (emphasis added). In other words, Judge Gilstrap wanted to nail down exactly what nonliteral elements of its software SAS was asserting against WPL. By doing so, Judge Gilstrap wanted to ensure that SAS had a legally viable theory of copyright infringement for the jury, and that WPL would have fair notice of it.

To survive the copyrightability hearing, SAS just needed to identify *some* protectable expression that it claimed WPL had copied. The jury would then decide whether and to what extent such copying had occurred. This was not a heightened or unfair burden; it is simply what the Fifth Circuit has explicitly stated is *required* for a plaintiff with a nonliteral software copyright infringement claim to survive summary judgment. *See Gen. Universal Sys.*, 379 F.3d at 143–44. Judge Gilstrap was well within his authority to require SAS to articulate a legally viable theory of copyright infringement prior to holding a jury trial. *See, e.g.*,

Savant Homes, Inc. v. Collins, 809 F.3d 1133 (10th Cir. 2016) (resolving copyrightability as a matter of law at summary judgment with the aid of expert testimony). Clarification of the legal scope of the plaintiff's rights and the plaintiff's theory of infringement are essential to fair and efficient litigation. *See* Part II.A *supra*. Here, years of litigation elapsed and huge amounts of money were spent without the plaintiff ever articulating a legally viable theory copyright infringement. This case demonstrates the need for particularity in nonliteral software copyright litigation, especially for those defendants who write code the world depends on but do not have huge sums of money to spend in its defense.

CONCLUSION

For the reasons set forth above, the district court's decision should be affirmed.

Dated: August 30, 2021

Respectfully submitted,

DURIE TANGRI LLP

By: */s/ Joseph C. Gratz*

Joseph C. Gratz
Samuel J. Zeitlin
Durie Tangri LLP
217 LEIDESDORFF STREET
SAN FRANCISCO, CA 94111
Telephone: 415-362-6666
Facsimile: 415-236-6300

Attorneys for *Amicus Curiae* GitHub, Inc.

CERTIFICATE OF COMPLIANCE WITH TYPE-VOLUME LIMITATION, TYPEFACE REQUIREMENTS, AND TYPE STYLE REQUIREMENTS

1. This brief complies with the type-volume limitation of Federal Rule of Appellate Procedure 32(a) or Federal Rule of Appellate Procedure 28.1. This brief contains 3,955 words, excluding the parts of the brief exempted by Federal Rule of Appellate Procedure 32(f).

2. This brief complies with the typeface requirements of Federal Rule of Appellate Procedure 32(a)(5) or Federal Rule of Appellate Procedure 28.1 and the type style requirements of Federal Rule of Appellate Procedure 32(a)(6). This brief has been prepared in a proportionally spaced typeface using Microsoft Word 2016 in Times New Roman, 14 point font.

Dated: August 30, 2021

Respectfully submitted,

DURIE TANGRI LLP

By: */s/ Joseph C. Gratz*

Joseph C. Gratz
Samuel J. Zeitlin
Durie Tangri LLP
217 LEIDESDORFF STREET
SAN FRANCISCO, CA 94111
Telephone: 415-362-6666
Facsimile: 415-236-6300

Attorneys for *Amicus Curiae* GitHub, Inc.

CERTIFICATE OF SERVICE

I hereby certify that on this 30th day of August, 2021, I caused the foregoing **BRIEF OF *AMICUS CURIAE* GITHUB, INC. IN SUPPORT OF DEFENDANT-APPELLEE AND AFFIRMANCE** to be served by electronic means via Court's CM/ECF system on all counsel registered to received electronic notices.

/s/ Joseph C. Gratz

Joseph C. Gratz
Durie Tangri LLP
217 LEIDESDORFF STREET
SAN FRANCISCO, CA 94111
Telephone: 415-362-6666
Facsimile: 415-236-6300

Attorneys for *Amicus Curiae* GitHub, Inc.