System for reproducible code environments - Dohrnii

Dimitar Mileski
Ss. Cyril and Methodius University
Faculty of Computer Science and Engineering
Skopje, North Macedonia
dimitar.mileski@ieee.org

Marjan Gusev
Ss. Cyril and Methodius University
Faculty of Computer Science and Engineering
Skopje, North Macedonia
marjan.gushev@finki.ukim.mk

***Abstract*—The reproducibility of code and the whole environment in which that code needs to be executed takes up much of the time of software engineers, computer scientists, and anyone who writes code. There are several reasons why we can not get the same results if the original data is available to us. This may be because the environment is not configured the same way as the source code author, we do not have enough knowledge of the technology used, poorly written documentation, lack of documentation, different versions of libraries, tools, development frameworks, mistakes made by one who wants to reproduce the code or there are errors in the code or errors that occurred after a certain period of time. In this paper, we introduce a new system for reproducible coding environments called Dohrnii, as a cloud-based solution. The main purpose is to save the time lost when we want to reproduce the results of a project for the first time. Every Dohrnii environment contains a video, description, instance (virtual machine), resources, and evaluation. This means that with this system it will be possible to reproduce environments in which the code needs to be executed, for that there will be a video where it will be shown how to reproduce the code, instance (virtual machine), description, additional resources, and part for an evaluation to determine if what is in the video corresponds to what is in the instance.**

***Index Terms*—code reproducibility, code environment, web platform, cloud computing**

I. Introduction

The reproducibility of the code is the process by which we will get the same results as the author of the code if we have access to the original data [1].

It takes time to configure and set up the environment when we try to reproduce code, to get the same results using the original data. In many cases, we spend a lot of time so that in the end we can not reproduce the results even though we have the original data. This is a big problem because software engineers, computer scientists, and anyone who writes code spends a lot of time trying to reproduce the results. Part of the job of software engineers and computer scientists is to try out different projects. Many hours and even days are spent trying to reproduce some code, which in the end may not be reproducible. There are several reasons why we can not get the same results as the author of the paper, or the project.

A. Code reproducibility problems

There are a number of errors that can lead to not reproducing the results. This is because the environment in which the code is executed is not the same as the environment in which the author of the code executed it and obtained the results. Common problems are identified by:

- not being familiar with technology,
- not having the opportunity to take the time to learn new technology,
- trying to see the end results, insufficient expertise in the field.

The problem is usually found in a new programming language, a new library, or a new software development framework. Extensive and complex documentation, poorly written documentation, and often incomplete or no documentation at all can be a big problem. Different versions of libraries, programming languages, and software development frameworks are one of the most common reproducibility issues. Software version incompatibility is often a problem. In order to solve this problem, the author of the source code needs to write text documentation write requirements, Docker File [2], or similar configuration files where the versions will be documented. Often package management tools do automatic version updates which can cause conflicts. The one who wants to reproduce the code also makes mistakes. In many cases, there are errors in the code from the start, and we spend time trying to reproduce the results from the code that is not working. Library support ends after a certain period, there are errors in the library or others problems with the libraries used. Some systems are not reproducible because they depend on a service that may have problems and cause errors. In all these cases, the computer scientist, the computer engineer, and all those who write the code will spend time without getting the same results as the author of the project.

B. Methods to reproduce the results

When we try to reproduce code we first see if the author of the code has documentation that we can follow. If it is not there or it is not complete, problems with reproducing the code are possible. In such cases, we can ask for help from the official documentation of the programming language, the library, and the development framework. More and more video platforms are used on which there are videos showing software development. Forums and blogs are also used so that we can solve a certain problem or replace part of the code. If the problem is in the environment in which the code is executed, which is initially set before the code is executed over the data set, configuration files, installations, tools, programs, integrated development environments, development frameworks, and libraries can solve the problem, and finally reproduce the code. If the code has documentation that lists the libraries used and their versions and all the details about the environment in which the code needs to be executed, we can reproduce the results.

For some projects, library version file, such as requirements.txt, package.json, pom.xml, and others is sufficient. Docker containers [2] make the environments in a container easily reproducible. Google Colab [3], and Jupyter Notebook

[4], are also platforms for reproducibility of code that is stored and executed in the cloud. The problem arises because the code and data are often not sufficient to obtain the same results as those obtained by the author. The environment in which the solution is developed is different from the environment in which we try to reproduce the results.

C. How the Dohrnii system solves the problem

The system for reproducible code environments (Dohrnii) tries to solve the problem with a unique combination of the previous methods. The system is based on Dohrnii environments. Each environment is composed of a video recorded by the author of the code using the screen recording tool on the Dohrnii platform. The video explains how to reproduce the results.

Short description of the Dohrnii environment, a description of the code, the data set, and the results to be obtained at the end. Each environment has link to the code repository. There is a list of additional resources. This section lists links to additional resources that are not in the code repository. It could be a link to a data set, a link to documentation, or some other useful link. All previous parts of the Dohrnii environment will help us to be able to reproduce the code in a virtual machine created by the Dohrnii system itself, and that virtual machine is in the cloud. SSH can be used to connect to the instance (virtual machine) [5] SSH Web client of choice or one proposed in Dohrnii system. If the instance has a graphical user interface, then we can connect to with Remote Desktop [6], Web or Desktop version of Remote Desktop [6]. Finally, in each Dohrnii environment, there is a section for evaluation. In the evaluation section, two answers are possible, the environment is reproducible or the environment is not reproducible, i.e. it means that what is in the video corresponds to the current state of the instance (virtual machine). This combination of screen recording, description, instance (virtual machine), code repository, additional resources, and evaluation section should provide a Web-based, cloud-based solution for reproducible code environments.

The paper is further organized as follows. An overview of similar systems for reproducible coding environments is given in Section II. The system is described in Section III with details on architecture, implementation technologies, and functional description. Section IV addresses the conclusions, including a discussion on advantages and disadvantages, and reveals a perspective on future work.

II. Related Work

This section will look at existing solutions that try to solve the problem of reproducibility. The Jupyter Project [4] is a project and community that aims to develop open-source software, open standards, and interactive computing services for multiple programming languages. The Jupyter project is made up of several subprojects such as Jupyter Notebooks [4], Jupyter Hub [7], Jupyter Lab [8]. Jupyter Notebook is an open-source web application that allows you to create and share documents with code, equations, visualizations, and text. Usage includes data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and more. Jupyter Notebooks if it is hosted can be a good platform for reproducibility. JupyterHub is a platform for hosting Jupyter Notebooks. It is used by students, scientists, and software developers.

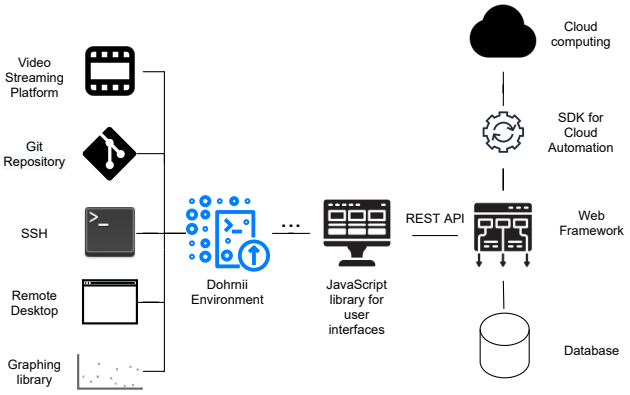

Fig. 1. System Architecture

Another project based on Jupyter is Binder [9]. Binder is an open-source web service that allows users to create interactive, reproducible cloud environments. Some of the technologies included in Binder are JupyterHub, Kubernetes [10], Repo2Docker [11]. Binder works with pre-existing community projects to create interactive versions of code repositories that exist on sites like GitHub [12].

Another cloud system based on Jupyter projects that provides computer resources such as GPU and TPU is Google Colab [13]. Collaboratory, or Colab for short, is a Google Research product that allows developers to write and execute Python code through their browser. These are Jupyter notebooks that do not require installation and have a free version that gives free access to Google's computer resources, such as GPUs and TPUs. Another such project is JupyterBook [14]. Jupyter Book is a collection of Jupyter notebooks.

There are also interactive environments for executing code in the field of Web development. Some of the client-side code development environments are:

- CodeSandbox is an online web development editor. CodeSandbox [15] is a tool for quickly building prototypes, experimenting, and sharing code with other developers. Used to create static Web pages, Web applications as a whole, or components of any Web browser device.
- JSFiddle [16] is an online service, integrated development environment, and community for testing and displaying user-generated code snippets. HTML, CSS, and JavaScript code are written and shared via JSFiddle. JSFiddle also enables the simulation of AJAX calls.
- CodePen [17] is a system for testing and displaying code snippets created by developers. It functions as a code editor and open-source learning environment, where developers can create snippets of code called "pens" and test them.

III. Description of the Dohrnii System

A. System architecture

Figure 1 shows the system architecture with the following components:

- **Dohrnii environment** - Dohrnii environment is an abstraction of all the technologies needed to have a reproducible project.
- **Video platform** - Screen recording and instructions on how to reproduce the environment are uploaded on a

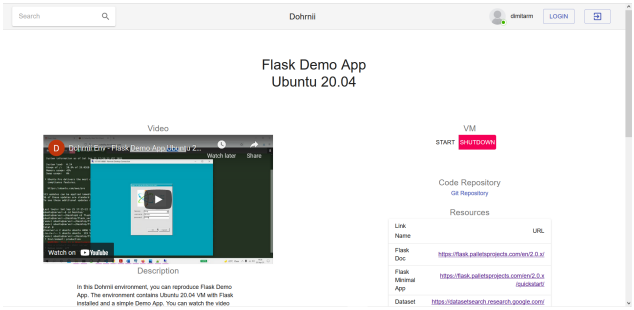

Fig. 2. Dohrnii environnment - video, description, VM controls, code repository, resources

video platform. These are external video platforms that are not part of Dohrnii.

- **Git repository** - Code is versioned with Git [18] and is hosted by external Git repository hosting services.
- **SSH** - SSH connections to instances (virtual machines) are made with Web-based SSH clients, which are free to use and open source.
- **Remote Desktop** - In order to be able to see the graphical user interface of the instances (virtual machines), we need to use the Remote Desktop client. SSH and Remote Desktop clients may not be Web-based, but Web-based RD clients are recommended to have a fully Web-based system.
- **Visualization Library** - Graph for the evaluation section of the Dohrnii environment is ploted with the help of visualization library.
- **Javascript library** - The client application is built with Javascript library, which uses the REST API [19] to perform basic CRUD operations, create, read, update and delete. This library includes other packages such as a screen capture package using the Screen Capture API [20], an icon package, and a Bootstrap [21] CSS framework.
- **Web framework** - The web framework should be compatible with the database. The choice of the framework also depends on the programming language in which the framework is written and the programming language of the library for cloud automation, starting and stopping instances (virtual machines) in the cloud.
- **Database** - The database stores user information so that we can implement authentication and authorization. Data for Dohrnii environments such as title, description, evaluation section data, link to Git repository, and links to additional resources are persisted in the database.
- **Cloud computing** - Cloud Instances (virtual machines) are created, started, stopped, and cloned for the users to reproduce the results.
- **Cloud computing automation** - The Automation Part uses a Software Development Kit (SDK) for automating cloud operations.

B. Implementation technologies

Technologies used for each of the components of the system architecture are given in this section. As shown in Figure 2, the Dohrnii environment contains title, video, description, instance (virtual machine) controls, code repository and resources.

- **Video platform** - We use YouTube for the video platform in the Dohrnii [22]. YouTube is a Google-owned video-sharing platform for the Internet and social media. The advantage of an existing platform is that users can monetize videos through existing platforms. The user who creates the Dohrnii environment should upload the screen recording with the code reproducibility instructions on the YouTube platform and insert the link to the YouTube video when creating the Dohrnii environment.
- **Git repository** - A link to GitHub is placed in each Dohrnii environment. One Dohrnii environment can have multiple code repositories. Other services such as GitLab [23] and BitBucket [24] can also be used.
- **Visualization Library** - To display the evaluation section shown in Figure 3 of the Dohrnii environment we will use the Plotly [25] library for data visualization. Plotly is an open-source library for data visualization. Plotly libraries are available for multiple programming languages including Python, Julia, R, ggplot2, MATLAB, and FSharp. We will use the plotly.js React component from Plotly to visualize the Dohrnii environment evaluation section.
- **Screen capture** - Figure 4 shows the screen capture interface for creating the Dohrnii environment. The Dohrnii screen capture uses a React js component called the React-media-recorder [26]. React-media-recorder is a React component that can be used to record audio, video, and screen. React-media-recorder uses the MediaRecorder API [27]. The MediaRecorder interface of the MediaStream recording API provides easy media recording functionality. Created using the MediaRecorder () constructor. An option that is important to us is that the recorded video is available for download.
- **Javascript library** - The React js [28] library is used to create the client application. React (also known as React.js or ReactJS) is a free, open-source JavaScript library for building user interfaces or components. React can be used as a basis for developing single-page applications or mobile applications. The most important React components we have used are React-media-recorder component and Material-UI. Material-UI [29] is a simple and customizable component library for building React applications.
- **Web framework** - We have used Django for the Web framework [30]. Django is a free web framework written in Python that follows the model-template-views (MTV)

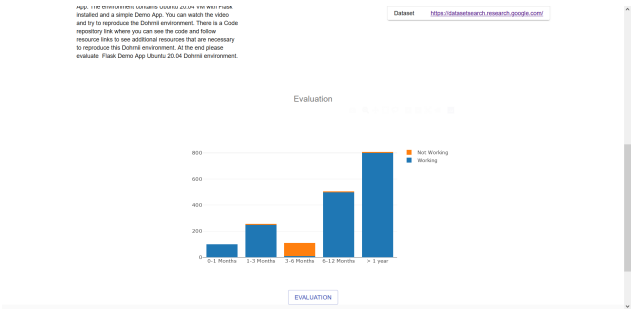

Fig. 3. Dohrnii environnment - evaluation

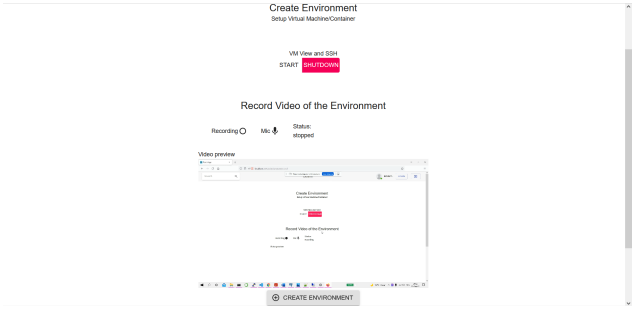

Fig. 4. Create Dohrnii environment view

architecture. The framework emphasizes the usability and ”connectivity” of the components, less code, fast development, and the principle of non-repetition. Python is used for settings, files, and data models. Django also provides an optional administrative interface for creating, reading, updating, and deleting dynamically generated data.

- **Database** - PostgreSQL is used as the database in the Dohrnii system [31]. Django has built-in support for PostgreSQL which facilitates the process of developing Django applications with PostgreSQL. PostgreSQL, also known as Postgres, is a free and open-source relational database management (RDBMS) that emphasizes scalability and is SQL compliant. PostgreSQL features transactions with properties of atomicity, consistency, isolation, endurance (ACID), automatic updates, materialized views, external keys, and stored procedures. It is designed to handle a wide range of workloads, from single machines to data warehouses or Web services with multiple concurrent users.
- **Cloud computing** - The instances that are created, started, and terminated on the Dohrnii system are AWS EC2 [32] instances. Amazon Elastic Compute Cloud (EC2) is part of Amazon.com’s cloud computing platform, Amazon Web Services (AWS), which allows users to rent virtual machines to run their own computer applications. EC2 encourages the scalable deployment of applications by providing a Web service through which the user can upload an Amazon Machine Image (AMI) to configure a virtual machine, which Amazon calls an ”instance” that contains the desired software. The user can create, start and terminate instances in the cloud.
- **Cloud computing automation** - The AWS SDK for

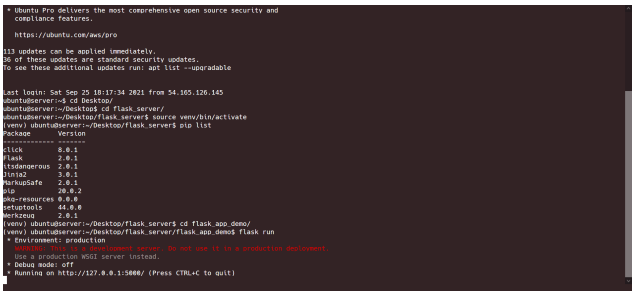

Fig. 5. SSH to Dohnrii environment

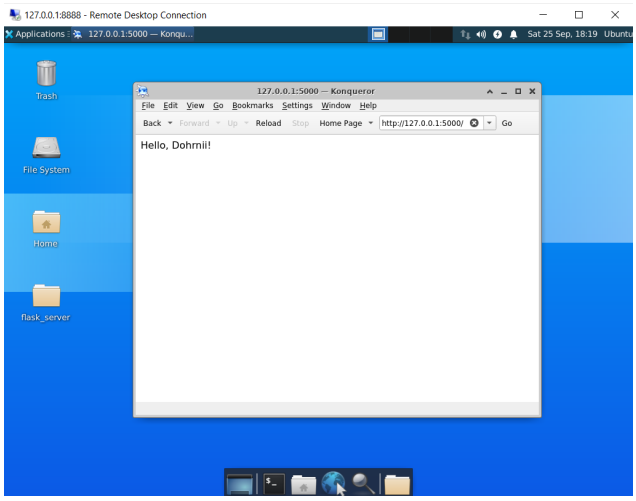

Fig. 6. Remote desktop conntecton to Dohrnii environment

Python (Boto3) [33] is used to create, configure, and manage AWS services such as Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Simple Storage Service (Amazon S3). The SDK provides an object-oriented API as well as low-level access to AWS services. Boto3 is a software development kit for Amazon Web Services (AWS) using the Python programming language, which allows Python developers to write software that uses services such as Amazon S3 and Amazon EC2. Listed below are just some of the features of the AWS SDK for Python (Boto3). Manage Amazon EC2 instances, get basic information about your Amazon EC2 instances, start and stop detailed instance monitoring of Amazon EC2, start and stop an instance of Amazon EC2, restart an instance of Amazon EC2, working with Amazon EC2 key pairs, get information about your key pairs, generate a pair of Amazon EC2 instant access keys, delete an existing pair of keys, working with security groups in Amazon EC2, get information about your security groups, create an Amazon EC2 instance access security group, delete an existing security group.

- **SSH** - Figure 5 shows Sshwifty, SSH and Telnet [34] client made for the Web. Can be deployed to our server to provide SSH and Telnet access interface for any compatible (standard) Web browser. Sshwifty is an open-source Web SSH client that allows you to connect to SSH on a remote machine without downloading additional software.
- **Remote Desktop** - Remote Desktop Protocol (RDP) is a commercial protocol developed by Microsoft that provides the user with a graphical interface for connecting to another computer over a network connection. The user in Figure 6 uses RDP client software for this purpose, while the instance (virtual machine) must be an RDP server. RDP client exist for most versions of Microsoft Windows (including Windows Mobile), Linux, Unix, macOS, iOS, Android, and other operating systems. RDP servers are built into Windows operating systems. An RDP server for Unix and OS X also exists. By default, the server listens on TCP port 3389 and UDP port 3389. To use RDP with Dohrnii environments it

is recommended to use Web RDP clients to have a completely web-based experience.

C. Functional description

The Dohrnii system is designed to be used by a variety of users who write code from developers, software developers, computer scientists, electrical engineers, scientists in other fields who write code.

Examples of use cases for the Dohrnii code reproducibility system include:

- Users can browse the system and reproduce the code in the environments. Developers who work in a company can share the environments so that they can reproduce a project. These could be projects in Web Development, Artificial Intelligence, or other areas.
- Professors, assistants, and laboratory assistants can insert link to the Dohrnii environment at the end of the presentation or exercise so that students can reproduce the results.
- Scientists can use Dohrnii environment as an open-source environment and enable others to reproduce the results. The link from the Dohrnii environment should be placed on the platforms for hosting scientific papers, or on the servers for the preprinted versions of the papers.
- Developers who ask questions on one of the software development question platforms, where the answer with the most votes to the question can be a link to the Dohrnii problem-solving environment or there is an environment that has a similar solution for the problem.
- Users can also access the Dohrnii system from existing code repositories with a link to the Dohrnii environment.
- Content creators on one of the video platforms that create videos in the field of computer science and computer engineering to put a link in the video description to the Dohrnii environment.

Use of the Dohrnii system requires basic knowledge and understanding of SSH, Remote Desktop Protocol, SSH Tunneling, and Port Forwarding.

IV. Conclusion

A new system for reproducible coding environments Dohrnii was introduced. The main purpose of the system is to save the time lost when we want to reproduce the results of a project.

The system was developed with sophisticated cloud-based architecture supported by modern technologies.

The advantages of the Dohrnii system are that it is a complete Web-based solution that uses cloud computing. All the tools needed to successfully reproduce an environment are available in the Web version. Such are screen capture, SSH, and Remote Desktop, clients.

The biggest disadvantage of the Dohrnii system is that it requires a large amount of resources. Because to reproduce the environment we use instances (virtual machines) that need working memory, storage, and processing resources. Disadvantages of the system include the disadvantages of virtual machines as a technology. The cost of creating and launching multi-user instances can be very high, depending on the cloud. The system is intended to be able to support the concurrent creation, start-up, and termination of instances for more users at the same time and the demand for the resources can be high, thus the cost of operating the system. The Dohrnii system now uses external SSH Web clients and external Web Remote Desktop clients. This is a potential security issue for the system as instance connection information will be shared with systems other than Dohrnii.

Possible improvements to the system include additional development including cloud automation with SDK Boto3. There would be a limit to the number of instances that can be run by a single user, we can also limit the time that each instance has from the time it is started in order to save resources. One of the priorities would be to have an SSH client built into the system. Additionally, SSH clients can be deployed supporting the free and open-source. Remote Desktop client built into the system to avoid using external clients. Consider possible ways to improve visualization in the Dohrnii environment evaluation section. Development of a built-in program for editing screen recording. That program would be Web-based and would have only basic features like trimming the screen recording and inserting text over the video itself if something needed to be further explained.

References

[1] V. Stodden, "The scientific method in practice: Reproducibility in the computational sciences," 2010.

[2] C. Anderson, "Docker [software engineering]," *Ieee Software*, vol. 32, no. 3, pp. 102–c3, 2015.

[3] M. J. Nelson and A. K. Hoover, "Notes on using google colaboratory in ai education," in *Proceedings of the 2020 ACM conference on innovation and Technology in Computer Science Education*, 2020, pp. 533–534.

[4] F. Pérez and B. E. Granger, "Ipython: a system for interactive scientific computing," *Computing in science & engineering*, vol. 9, no. 3, pp. 21–29, 2007.

[5] T. Ylonen, "Ssh–secure login connections over the internet," in *Proceedings of the 6th USENIX Security Symposium*, vol. 37, 1996.

[6] R. Rahim, J. Simarmata, A. Purba, M. A. Prayogi, A. Sapta, O. Krianto, and S. Suharman, "Internet based remote desktop using indy and socket component," *Int. J. Eng. Technol*, vol. 7, no. 2.9, pp. 44–47, 2018.

[7] L. Fernández, R. Andersson, H. Hagenrud, T. Korhonen, E. Laface, B. Zupanc *et al.*, "Jupyterhub at the ess. an interactive python computing environment for scientists and engineers," in *This conference*, 2016.

[8] B. Granger and J. Grout, "Jupyterlab: Building blocks for interactive computing," *Slides of presentation made at SciPy*, vol. 2016, 2016.

[9] B. Ragan-Kelley and C. Willing, "Binder 2.0-reproducible, interactive, sharable environments for science at scale," in *Proceedings of the 17th Python in Science Conference (F. Akici, D. Lippa, D. Niederhut, and M. Pacer, eds.)*, 2018, pp. 113–120.

[10] D. Bernstein, "Containers and cloud: From lxc to docker to kubernetes," *IEEE Cloud Computing*, vol. 1, no. 3, pp. 81–84, 2014.

[11] J. Forde, T. Head, C. Holdgraf, Y. Panda, G. Nalvarete, B. Ragan-Kelley, and E. Sundell, "Reproducible research environments with repo2docker," 2018.

[12] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, "Social coding in github: transparency and collaboration in an open software repository," in *Proceedings of the ACM 2012 conference on computer supported cooperative work*, 2012, pp. 1277–1286.

[13] T. Carneiro, R. V. M. Da Nóbrega, T. Nepomuceno, G.-B. Bian, V. H. C. De Albuquerque, and P. P. Reboucas Filho, "Performance analysis of google colaboratory as a tool for accelerating deep learning applications," *IEEE Access*, vol. 6, pp. 61 677–61 685, 2018.

[14] D. Mileski, M. Jovanovik, and D. Trajanov, "Semantic web and data science integration using computational books," 2021.

[15] "Codesandbox, together." [Online]. Available: https://codesandbox.io/

[16] JSFiddle. [Online]. Available: https://jsfiddle.net/

[17] "Codepen," Feb 2022. [Online]. Available: https://en.wikipedia.org/wiki/CodePen

[18] S. Chacon and B. Straub, *Pro git*. Springer Nature, 2014.

[19] M. Masse, *REST API design rulebook: designing consistent RESTful web service interfaces*. " O'Reilly Media, Inc.", 2011.

[20] "Screen capture api - web apis: Mdn." [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Screen_Capture_API

[21] J. Spurlock, *Bootstrap: responsive web development*. " O'Reilly Media, Inc.", 2013.

[22] J. Burgess, "Youtube," in *Oxford Bibliographies Online*, L. H. Meyer, Ed. United Kingdom: Oxford University Press, 2011, pp. 1–1. [Online]. Available: https://eprints.qut.edu.au/46719/

[23] J. M. Hethey, *GitLab Repository Management*. Packt Publishing, 2013.
[24] Atlassian, "The git solution for professional teams." [Online]. Available: https://bitbucket.org/product
[25] J. M. Perkel *et al.*, "Data visualization tools drive interactivity and reproducibility in online publishing," *Nature*, vol. 554, no. 7690, pp. 133–134, 2018.
[26] "React-media-recorder." [Online]. Available: https://www.npmjs.com/package/react-media-recorder
[27] "Mediarecorder - web apis: Mdn." [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/MediaRecorder
[28] E. Elrom, "React router and material-ui," in *React and Libraries*. Springer, 2021, pp. 79–113.
[29] "The react component library you always wanted." [Online]. Available: https://mui.com/
[30] A. Holovaty and J. Kaplan-Moss, *The definitive guide to Django: Web development done right*. Apress, 2009.
[31] B. Momjian, *PostgreSQL: introduction and concepts*. Addison-Wesley New York, 2001, vol. 192.
[32] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "A performance analysis of ec2 cloud computing services for scientific computing," in *International Conference on Cloud Computing*. Springer, 2009, pp. 115–131.
[33] "Boto3 documentation." [Online]. Available: https://boto3.amazonaws.com/v1/documentation/api/latest/index.html
[34] J. Postel, "Telnet protocol specification," in *RFC 854, ISI*. Citeseer, 1983.