

**SCTR’s Pune Institute of Computer Technology , , Dhankawadi, Pune**

**AN INTERNSHIP REPORT ON**

Inhouse Project

**SUBMITTED BY**

Name: Ojasvi Deshpande

Class: TE 02

Roll no: 31221

**Under the guidance of**

Dr. K. C. Waghmare

**DEPARTMENT OF COMPUTER ENGINEERING**

ACADEMIC YEAR 2022-23



**DEPARTMENT OF COMPUTER ENGINEERING**

**SCTR’s Pune Institute of Computer Technology Dhankawadi, Pune**

**Maharashtra 411043**

**CERTIFICATE**

This is to certify that the SPPU Curriculum-based internship report entitled

**“Inhouse Project”**

Submitted by

Ojasvi Deshpande Roll no: 31221

has satisfactorily completed the curriculum-based internship under the guidance of Dr. K. C. Waghmare towards the partial fulfillment of third year Information Technology Semester VI, Academic Year 2022-23 of Savitribai Phule Pune University.

**Dr. K. C. Waghmare Dr. G. V. Kale**

Internship Guide Head

PICT, Pune Department of Computer

Engineering, PICT, Pune

Place: PICT, Pune

Date: 15/05/2023

**ACKNOWLEDGEMENT**

It gives me great pleasure in presenting the internship report on Inhouse Project. First of all I would like to take this opportunity to thank my internship guide Dr. K.C. Waghmare for giving me all the help and guidance needed. I am really grateful for his kind support and valuable suggestions that proved to be beneficial in the overall completion of this internship. I am thankful to our Head of Computer Engineering Department, Dr. G.V.Kale, for her indispensable support and suggestions throughout the internship work. I would also genuinely like to express my gratitude to the Department Internship Coordinator, Prof. P. P. Joshi, for her constant guidance and support and for the timely resolution of the doubts related to the internship process. Finally, I would like to thank my mentor, Prof. D.D. Kadam for her constant help and support during the overall internship process.

Ojasvi Deshpande

31221

**Contents**

1. [Title 3](#_TOC_250012)
2. [Introduction 3](#_TOC_250011)
3. [Problem Statement 4](#_TOC_250010)
4. [Objectives and Scope 4](#_TOC_250009)

[4.1 Objectives . . . . . . . . . . . . . . . . . . . . . . . . 4](#_TOC_250008)

[4.2 Scope . . . . . . . . . . . . . . . . . . . . . . . . . . 5](#_TOC_250007)

1. [Methodological Details 6](#_TOC_250006)
   1. [Designing and Developing Website . . . . . . . . . . 6](#_TOC_250005)
2. [Modern engineering tools used 7](#_TOC_250003)
3. [Outcome/ results of internship work (screenshots of work done) 9](#_TOC_250002)
4. [Conclusion 12](#_TOC_250001)
5. [References 13](#_TOC_250000)

# Title

Web application to get Hindex and i10 index of authors' publications from google scholar.

# Introduction

The web application developed aims to provide the h-index and i10 index of authors' publications by retrieving data from Google Scholar. The application will have a user interface where users can enter the name of the author. It will then scrape the data from the platform and fetch the required information.

To accomplish this, we set up a web server using NodeJs language and its libraries. The server will handle HTTP requests and responses for the web application. Users will interact with the application through a user interface, where they can input the author's name and get a report of the information in the form of pdf.

The application will integrate with Google Scholar by querying the user requirements. The retrieved publication data, including citation counts for each paper, will be used to calculate the h-index and i10 index. The results will be displayed to the user along with other filters.

To ensure a smooth user experience, error handling mechanisms will be implemented to address cases where the author's profile is not found.

Once the development is complete, the web application can be deployed to a web server or cloud platform to make it accessible to users. Regular maintenance and updates would be performed to accommodate any changes in the website structures of Google Scholar.

# Problem Statement

Develop a web application to get hindex and I index of authors’ publications from google scholar.

# Objectives and Scope

## Objectives

* **Design the user interface:** Create a web page with a form where users can enter the author's name or identifier (e.g., Google Scholar profile URL) to retrieve their publication metrics. You can use HTML, CSS, and JavaScript to design and structure the UI.
* **Handle form submission:** Implement JavaScript code to handle form submission. Capture the input from the form and initiate the process of retrieving publication metrics from Google Scholar.
* **Scrape Google Scholar data:** Use a library like Puppeteer (as mentioned earlier) to programmatically control a headless Chrome browser and scrape the necessary data from the Google Scholar website. With Puppeteer, you can search for the author's profile, extract the required information, and navigate through the pages to obtain publication details.
* **Process the scraped data:** Once you retrieve the author's publication data, parse and process it to calculate the h-index and i10-index. The h-index represents the author's highest number of papers that have received at least h citations, while the i10-index represents the number of papers with at least 10 citations.
* **Display the results:** Update the web page with the calculated h-index and i10-index values. You can dynamically update the UI using JavaScript to show the results to the user.
* **Test and refine:** Thoroughly test your web application to ensure it functions correctly. Verify that it retrieves accurate publication metrics and handles different scenarios gracefully. Refine and optimize the code as needed.

## Scope

The scope of this project has been to understand the requirement of the web application and the idea behind it. The aim of the project was to develop a website that can retrieve the h-index and i10 index of authors' publications from Google Scholar. The application will have a user interface where users can input the author's, and the application will retrieve the relevant information by scraping data from the websites. The project will involve setting up a web server and integrating with the Google Scholar. The application will be programmed using NodeJs programming language and relevant libraries such as puppeteer.

# Methodological Details

## Designing and Developing Website :

## Steps to start a Node Project :

* Set up Node.js: Ensure that you have Node.js installed on your system. You can download the latest version from the official Node.js website (https://nodejs.org). Follow the installation instructions for your operating system.
* Create a project directory: Choose a directory where you want to create your Node.js project. Open a terminal or command prompt, navigate to the desired directory, and create a new folder for your project.
* Initialize a new Node.js project: Use the npm init command to initialize a new Node.js project in your project directory. This command will guide you through a series of prompts to set up your project's details, such as package name, version, entry point, dependencies, etc.
* Install dependencies: If your project requires any third-party dependencies or libraries, you can use the npm install command to install them. For example, if you want to install Express.js as a dependency, you would run: npm install express.
* This command will download and install Express.js and its dependencies in a folder named node\_modules within your project directory.
* Create your application code: Start writing your Node.js application code. The entry point of your application is typically a JavaScript file, commonly named index.js or app.js. You can create this file in your project directory and start coding your Node.js application logic.
* Run your Node.js application: To run your Node.js application, execute the main JavaScript file using the node command.

**About puppeteer** :

Puppeteer is a Node.js library developed by the Chrome team at Google. It provides a high-level API for automating and controlling headless versions of the Chrome or Chromium browsers. With Puppeteer, you can programmatically interact with web pages, perform various tasks such as generating screenshots and PDFs, scraping data, running automated tests, and more.

**Steps to use puppeteer :**

* Install Puppeteer: Install the Puppeteer library as a dependency for your project using the npm install command. Run the following command in your project directory. This command will download and install Puppeteer and its dependencies in the node\_modules folder of your project.
* Import Puppeteer: In your JavaScript file, import Puppeteer using the require or import statement.
* Use Puppeteer: With Puppeteer imported, you can now use its API to control and interact with headless Chrome or Chromium. Here's a simple example that launches a browser, opens a new page, navigates to a website, and takes a screenshot:

const puppeteer = require('puppeteer');

async function run() {

const browser = await puppeteer.launch();

const page = await browser.newPage();

await page.goto('https://example.com');

await page.screenshot({ path: 'screenshot.png' });

await browser.close();

}

run();

In this example, puppeteer.launch() launches a headless browser instance, browser.newPage() creates a new page, page.goto() navigates to the specified URL, and page.screenshot() captures a screenshot of the page. You can explore Puppeteer's API documentation (https://pptr.dev/) to discover more features and methods for web automation, page manipulation, network interception, and other tasks. Run the Node.js script.

* By following these steps, you can start using the Puppeteer library in your Node.js project to automate browser tasks, scrape data, generate screenshots or PDFs, and perform various other web-related operations.

# Modern engineering tools used

1. **HTML:**

HyperText Markup Language, or HTML. With the aid of a markup language, it is used to design web pages. Markup lan- guage and hypertext are combined to create HTML. The link between web pages is defined by hypertext. The text docu- ment inside the tag that specifies the structure of web pages is defined using a markup language. This language is used to annotate (add notes to) material so that a computer can com- prehend it and modify the content as necessary. Most markup languages, like HTML, can be read by people. The language employs tags to specify what text processing is required.

1. **CSS :**

CSS, or cascading style sheets, is an acronym. It is a language for creating style sheets that describe the layout and appear- ance of markup-language documents. It gives HTML an addi- tional feature. Typically, it works with HTML to modify the look and feel of online pages and user interfaces. Any XML document type, including plain XML, SVG, and XUL, can be used with it.To construct user interfaces for web apps and many mobile applications, most websites combine CSS, HTML, and JavaScript.

1. **JavaScript:**

JavaScript, commonly referred to as the scripting language for websites, is a simple, single-threaded, interpreted, and cross- platform computer language. Many non-browser environments also use it, and it is widely accepted for web page develop- ment. JavaScript is a dynamically typed, weakly typed lan- guage. Both client-side and server-side development can be done with JavaScript. Both imperative and declarative lan- guages can be used with JavaScript.

1. **NodeJS:**

Node.js is an open-source, server-side runtime environment that allows developers to build and run JavaScript applications outside of a web browser. It uses the V8 JavaScript engine, developed by Google, which compiles JavaScript code into machine code, making it highly efficient.

Here are some key features and concepts associated with Node.js:

1. **Asynchronous programming:** Node.js is known for its ability to handle a large number of concurrent connections efficiently by using an event-driven, non-blocking I/O model. This means that instead of waiting for one operation to complete before moving to the next, Node.js can initiate multiple operations and process them in parallel, resulting in high performance and scalability.
2. **NPM (Node Package Manager):** NPM is the default package manager for Node.js, allowing developers to easily manage and install third-party libraries and modules needed for their projects. It provides access to a vast ecosystem of reusable modules, making it easier to build applications by leveraging existing code.
3. **Cross-platform Compatibility:** Node.js is available on multiple platforms, including Windows, macOS, and various Linux distributions. This cross-platform compatibility allows developers to write applications on one platform and easily deploy them on different environments.

# Outcome/ results of internship work (screen- shots of work done)

# Conclusion

In conclusion, the above project aims to develop a web application that retrieves the h-index and i10 index of authors' publications from Google Scholar. The project's objective is to provide researchers, academics, and interested users with an efficient and user-friendly tool to access these important metrics.

The project's scope includes setting up a web server, integrating with the Google Scholar and designing a user interface for inputting the author's name. The application retrieves the necessary data, calculates the h-index and i10 index, and presents the results along with a report to the user.

Node.js, a powerful JavaScript runtime environment, was used to develop the web application. Its asynchronous and non-blocking nature, along with the extensive ecosystem of packages offered by NPM, make it suitable for handling the required integrations and providing a fast and scalable application.

Overall, the project provides a valuable tool for retrieving and analyzing the h-index and i10 index of authors' publications from Google Scholar. It empowers users with insights into an author's scholarly impact and facilitates informed decision-making in the academic and research domains.

# References

1. [http://www.w3schools.com](http://www.w3schools.com/)
2. h[ttps://www.ja](http://www.javatpoint.com/javascript-tutorial)vatpoin[t.com/javascript-tutorial](http://www.javatpoint.com/javascript-tutorial)
3. <http://www.w3.org/standards/semanticweb/>
4. http://www.youtube.com
5. http://www.scholar.google.com
6. https://drive.google.com/drive/u/0/folders/1POKm4t71uTgsnd9PYo8eD9RsOkwvL4\_T
7. https://nodejs.dev/en/learn/
8. https://npmjs.com