

# ScholariZer

A Scholar Profile Analyzer

*Dominik Jentsch<sup>1</sup>, Tim Wolk<sup>2</sup>,  
Alexander Möhring<sup>3</sup>, Pablo Schmeiser<sup>4</sup>,  
Adham Gouda<sup>5</sup>*



---

<sup>1</sup>Phase-leader for Definition of Requirements

<sup>2</sup>Phase-leader for Design

<sup>3</sup>Phase-leader for Implementation

<sup>4</sup>Phase-leader for Quality Assurance

<sup>5</sup>Phase-leader for Final Presentation

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
	<b>Acronyms</b>	<b>5</b>
	<b>Definition of Terms</b>	<b>5</b>
<b>2</b>	<b>Software data/structure</b>	<b>9</b>
<b>3</b>	<b>Product environment</b>	<b>10</b>
3.1	Environmental requirements . . . . .	11
<b>4</b>	<b>Scope</b>	<b>12</b>
4.1	Justification . . . . .	12
4.2	Target group . . . . .	12
<b>5</b>	<b>General system functionalities/modules</b>	<b>13</b>
5.1	Mandatory requirements . . . . .	13
5.2	Optional requirements . . . . .	13
5.3	Delimitation criteria . . . . .	14
<b>6</b>	<b>Functional requirements</b>	<b>15</b>
6.1	Mandatory . . . . .	15
6.2	Optional . . . . .	19
<b>7</b>	<b>Non-functional requirements</b>	<b>21</b>
7.1	Mandatory . . . . .	21
7.1.1	Usability . . . . .	21
7.1.2	Security . . . . .	21
7.1.3	Performance . . . . .	21
7.1.4	Maintainability . . . . .	21
7.1.5	Scalability . . . . .	21
7.1.6	Data . . . . .	21
7.2	Optional . . . . .	22
7.2.1	Usability . . . . .	22
7.2.2	Security . . . . .	22
7.2.3	Scalability . . . . .	22
<b>8</b>	<b>Test cases</b>	<b>23</b>
<b>9</b>	<b>Scenarios</b>	<b>29</b>
9.1	Scenarios . . . . .	29
9.2	GUI mockups . . . . .	31
<b>10</b>	<b>System models</b>	<b>45</b>



# 1 Introduction

Researching scholars is widely used by a lot of people working in scientific fields and is essential for scientific researches.

ScholariZer solves the problem of transparency caused by the lack of differentiation between citation and self-citation. Furthermore scholariZer aims to provide a better and more transparent overview with differentiated and valuable metrics.

ScholariZer is meant for all the people that want to do a well-founded research without misleading numbers of alleged citations by the authors to push their documents.

ScholariZer aims to protect and improve the integrity and transparency of scientific research.

## Acronyms

**API** Application Programming Interface. 9, 20, 47

**DOI** Digital Object Identifier. 16

**GUI** Graphical User Interface. 13, 17, 18, 21, 47

**IDE** Integrated Development Enviroment. 47

**SQL** Structured Query Language. 9

**UML** Unified Modelling Language. 6, 47

## Terms

**Apache maven** A build automation tool used primarily for Java projects, hosted by the Apache Software Foundation. 21, 47

**author** Writer of a publication. 13, 18, 19, 23, 46

**Babel** A free and open-source JavaScript transcompiler that is mainly used to convert ECMAScript 2015+ (ES6+) code into backwards-compatible JavaScript code that can be run by older JavaScript engines or to transcompile TypeScript into JavaScript. 47

**Checkstyle** Checkstyle is a static code analysis tool used in software development for checking if Java source code is compliant with specified coding rules. 47

**Docker** A set of platform as a service (PaaS) products that use OS-level virtualization to deliver software in packages called containers. 10, 47

**ESLint** ESLint is a static code analysis tool for identifying problematic patterns found in JavaScript code. 47

**Figma** A collaborative web application for interface design. 47

**Git** A free and open source software for distributed version control. 47

**GitHub** An Internet hosting service for software development and version control using Git. 47

**Google Calendar** A time-management and scheduling calendar service developed by Google. 47

**Google Chrome** A cross-platform web browser developed by Google. 47

**guest** Someone who has no Profile or is not logged-in. 13, 15–20, 23, 24

**h-index** Metric for judging the quality of a researcher’s work based on the number of citations. 13, 18, 19

**i10-index** Metric for judging the quality of a researcher’s work based on the number of citations. 18

**IEEE Xplore digital library** A research database for discovery and access to journal articles, conference proceedings, technical standards, and related materials on computer science, electrical engineering and electronics, and allied fields. 47

**IntelliJ IDEA** An IDE for Java virtual machine–based languages such as Java, Groovy, Kotlin, and Scala. 47

**Java** A high-level, class-based, object-oriented programming language. 21, 47

**JavaDoc** A documentation generator for the Java language for generating API documentation in HTML format from Java source code. 47

**Jest** A JavaScript testing framework designed and built with a focus on simplicity and support for large web applications. It works with projects using Babel, TypeScript, Node.js and React. Jest doesn’t require a lot of configuration for first-time users of a testing framework. 47

**JUnit** A unit testing framework for the Java programming language. 47

**kubernetes** An open-source container orchestration system for automating software deployment, scaling, and management. 47

**LaTeX** A software system for document preparation. The writer uses markup tagging conventions to define the general structure of a document to stylise text throughout a document and to add citations and cross-references. 47

**Lucidchart** A web-based diagramming application that allows easy creation of UML diagrams. 47

**Markdown** A lightweight markup language for creating formatted text using a plain-text editor. 47

**metric** System of measurement, in the context of this project: h-Index, i10-Index, total citation count, papers published, all offered with and without self citations.. 15

**Microsoft Visio** A diagramming and vector graphics application that is part of the Microsoft Office family. 47

**Model-View-Controller** Software architectural pattern, commonly used for developing user interfaces that divide the related program logic into three interconnected elements. 9

**modular design** A design principle that subdivides a system into smaller parts called modules, which can be independently created, modified, replaced, or exchanged. 21

**Mozilla Firefox** A free and open-source web browser developed by the Mozilla Foundation. 47

**Next.js** An open-source web development framework enabling React-based web applications with server-side rendering and generating static websites. 47

**Notion** A note taking software to help capture thoughts, organize and manage projects. 47

**npm** A package manager for the JavaScript programming language. npm is the default package manager for the JavaScript runtime environment Node.js. 47

**Overleaf** A collaborative cloud-based LaTeX editor used for writing, editing and publishing scientific documents. 47

**paper** A scientific document regarding research of some kind. 9

**plugin** A software component that adds a specific feature to an existing program. 22

**PostgreSQL** A free and open-source relational database management system (RDBMS) emphasizing extensibility and SQL compliance. 47

**publication** Information or stories available to people in printed or electronic form. 5, 13–21, 23, 45, 47

**React** A free and open-source front-end JavaScript library for building user interfaces based on UI components. 21, 47

**recommended** A proposal, done by scholariZer, based on bookmarked papers, custom set preferences and followed authors.. 13, 19

**regular-citation** A citation which is not made by the author of the publication. 13, 18

**relevant** Of high importance, decided by scholariZer, based on a combination of metrics.. 15

**researcher** Someone who has published at least one publication. 9, 13, 14, 18–20, 39

- self-citation** If a researcher cites his own publication. 13, 15, 17–19
- Semantic Scholar** An artificial intelligence–powered research tool for scientific literature. 47
- SonarLint** SonarQube integrates with IDEs like IntelliJ IDEA and WebStorm through the SonarLint plug-ins, and also integrates with external tools like GitHub, and others. 47
- SonarQube** An open-source platform developed by SonarSource for continuous inspection of code quality to perform automatic reviews with static analysis of code to detect bugs and code smells on 29 programming languages. 47
- Spring** An application framework and inversion of control container for the Java platform. 47
- Spring Boot** Spring’s convention-over-configuration solution for creating stand-alone, production-grade Spring-based Applications that you can "just run". 47
- TypeScript** A superset of the programming language JavaScript, developed and maintained by Microsoft. 47
- UMLet** A free, open-source UML tool with a simple user interface. 47
- user** Someone who has a profile and is logged-in. 9, 13–21, 23, 24, 45–47
- WebStorm** An IDE for web, JavaScript and TypeScript development. 47



## 2 Software data/structure

The Software will be separated into frontend, backend and databases, as shown in figure 1. The Software will be built utilizing the Model-View-Controller design pattern, where View represents the frontend (i.e. the website the user sees and interacts with), while Controller and Model are in the backend communicating with databases and formatting the received/calculated data for the view to be able to display it to the user. A typical interaction consists of the user clicking on something on the web page. The View then sends a request containing all the available and needed information to the Controller, which itself then redirects this request to the appropriate part of the Model. The Model then requests data (if necessary) from either the IEEEExplore database, via an API call, or the profile database, which will contain all Information of the user. Once the Model has received it's needed data (if any) it modifies and formats it and returns it to the Controller once it's done. The Controller then passes the received information on to the View, which displays it to the user.

The only data that needs to be stored long term are the User profiles, which will be stored in SQL database.

For each registered user, the database will need to store:

- Login credentials (namely E-Mail and Password)
- User interests as selected by the user
- A List of researchers this user follows
- A List of papers this User has bookmarked

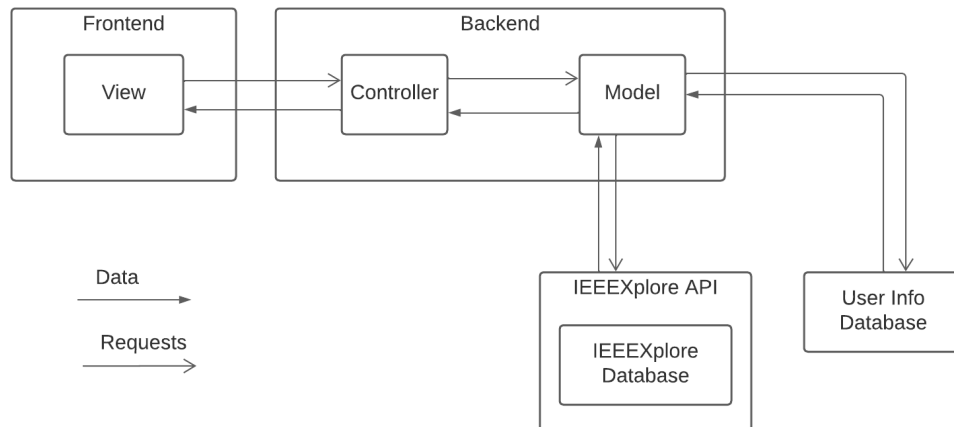


Figure 1: Architecture sketch

### 3 Product environment

A product environment is where the latest versions of software, products, or updates are pushed live to the intended users.

#### Server specifications:

- **Hardware:**

**These specifications are for 25 concurrent users with average requests per second frequency:**

Processor	Intel Xeon E5440 @ 2.83GHz
Memory	4 GB
Disk space	ca. 20 GB (3 GB for database files + enough for attachments)
Other	Network card is required

**This table is a simplified summary of higher usage levels of the server solution:**

Users	Threads / vCPU	RAM	HDD
50	8	12GB	40GB
100	12	32GB	60GB
200	24	64GB	200GB
500	24	128GB	500GB

- **Software:**

The server has to at least run a modern Linux based operating system. Since we are planning to use Docker to deploy the finished product to the server, there isn't really any hard software requirements for the server, other than running Linux.

### 3.1 Environmental requirements

- **Client side specifications:**

Scholarizer is ensured to run on systems with following specifications, lower specifications may work as well

	<b>Windows Requirements</b>	<b>Mac Requirements</b>	<b>Linux Requirements</b>
<b>Operating System</b>	Windows 8 or later	macOS Sierra 10.12 or later 64-bit	Ubuntu 14.04+, Debian 8+, openSUSE 13.3+, Fedora Linux 24+
<b>Processor</b>	Intel Pentium 4 or later	Intel	Intel Pentium 4 or later
<b>Memory</b>	2 GB minimum, 4 GB recommended		
<b>Screen resolution</b>	1280x1024 or larger		
<b>Application window size</b>	1024x680 or larger		
<b>Internet connection</b>	Required		

## **4 Scope**

### **4.1 Justification**

ScholariZer will be a user-friendly web application that helps ease the process of scientific research and education through granting access to research papers published worldwide.

### **4.2 Target group**

The target group are people interested in scientific research, including teachers, students and scientists.

## 5 General system functionalities/modules

### 5.1 Mandatory requirements

- ⟨MR10⟩ Guests can register and set up a profile.
- ⟨MR20⟩ Users and guests can search for publications and authors.
- ⟨MR30⟩ Users and guests can see how many times a publication has been cited.
- ⟨MR40⟩ Scholarizer offers a Graphical User Interface (GUI). (see Mockups)
- ⟨MR50⟩ Scholarizer differentiates between regular-citations and self-citations.
- ⟨MR60⟩ Users and guests are able to compare multiple researchers.
- ⟨MR70⟩ Scholarizer offers a recommended page to its users.
- ⟨MR80⟩ In order to read or download a publication, the user is redirected to an external web page.
- ⟨MR90⟩ Frequent co-authors are shown in a researcher's profile.
- ⟨MR100⟩ A tab is shown on an Author's profile displaying their peers who cited them the most.
- ⟨MR110⟩ The h-index and number of citations of a researcher are displayed with and without self-citations on their profile.

### 5.2 Optional requirements

- ⟨OR10⟩ Users and guests should be able to export references in plain text or BibTeX format.
- ⟨OR20⟩ Users should be able to bookmark publications.
- ⟨OR30⟩ Users should be able to follow researchers.
- ⟨OR40⟩ There should be an option to exclude co-authors from ⟨MR100⟩
- ⟨OR50⟩ Users should get a list of recommended researchers depending on their interests.
- ⟨OR60⟩ The relative impact score of a publication or researcher should be displayed.

### 5.3 Delimitation criteria

- $\langle D10 \rangle$  Scholarizer will not provide a feature to publish publications.
- $\langle D20 \rangle$  Scholarizer will not provide a feature to read publications.
- $\langle D30 \rangle$  Scholarizer will not provide a feature to download publications.
- $\langle D40 \rangle$  Scholarizer will not provide an option to rate publications or researchers.
- $\langle D50 \rangle$  User profiles managed by scholarizer will not be open to the public.

## 6 Functional requirements

### 6.1 Mandatory

1. Users can register and set up a user profile. ⟨MR10⟩
  - ⟨FM110⟩ The profile will be used to save data.
  - ⟨FM120⟩ Users can change their password.
  - ⟨FM130⟩ Users can change their email.
  - ⟨FM140⟩ Users can add interests
  - ⟨FM150⟩ Users can delete their profiles.
2. Users and guests can search for publications and authors. ⟨MR20⟩
  - ⟨FM210⟩ A search request for an author results in a preview of their profile and a list of all his publications.
  - ⟨FM220⟩ A search request for a publication results in a list of the most relevant publications, displayed to the user or guest.
    - ⟨FM220A⟩ The list of the most relevant publications has a length of 200, if there are enough publications to match the search parameters.
    - ⟨FM220B⟩ The list of the most relevant publications can be sorted.
      - ⟨FM220B1⟩ The list of the most relevant publications can be sorted ascending and descending by article number.
      - ⟨FM220B2⟩ The list of the most relevant publications can be sorted ascending and descending alphabetically by publication title.
      - ⟨FM220B3⟩ The list of the most relevant publications can be sorted by publication metrics.
  - ⟨FM230⟩ Users and guests can preview information about given publications.
    - ⟨FM230A⟩ Users and guests can preview the title of a given publication.
    - ⟨FM230B⟩ Users and guests can preview the author of a given publication.
    - ⟨FM230C⟩ Users and guests can preview the submission date of a given publication.
    - ⟨FM230D⟩ Users and guests can preview metrics regarding a given publication.
      - ⟨FM230D1⟩ Users and guests can see h-Index of a publication including and excluding self-citations.

- ⟨FM240⟩ searches all publications available in the IEEE Xplore API.
- ⟨FM250⟩ Users and guests can enter search terms which will be matched in accordance with various parameters.
- ⟨FM250A⟩ Users and guests can enter search terms which will be matched in accordance with free text.
  - ⟨FM250B⟩ Users and guests can enter search terms which will be matched in accordance with research field.
  - ⟨FM250C⟩ Users and guests can enter search terms which will be matched in accordance with article title.
  - ⟨FM250D⟩ Users and guests can enter search terms which will be matched in accordance with author.
  - ⟨FM250E⟩ Users and guests can enter search terms which will be matched in accordance with publisher.
  - ⟨FM250F⟩ Users and guests can enter search terms which will be matched in accordance with article number.
  - ⟨FM250G⟩ Users and guests can enter search terms which will be matched in accordance with publication title.
  - ⟨FM250H⟩ Users and guests can enter search terms which will be matched in accordance with publication year.
  - ⟨FM250I⟩ Users and guests can enter search terms which will be matched in accordance with affiliation.
  - ⟨FM250J⟩ Users and guests can enter search terms which will be matched in accordance with DOI.
  - ⟨FM250K⟩ Users and guests can enter search terms which will be matched in accordance with D-AU.
  - ⟨FM250L⟩ Users and guests can enter search terms which will be matched in accordance with D-publisher.
  - ⟨FM250M⟩ Users and guests can enter search terms which will be matched in accordance with D-pubtype.
  - ⟨FM250N⟩ Users and guests can enter search terms which will be matched in accordance with D-year.
  - ⟨FM250O⟩ Users and guests can enter search terms which will be matched in accordance with index terms.
  - ⟨FM250P⟩ Users and guests can enter search terms which will be matched in accordance with ISBN.



- ⟨FM250Q⟩ Users and guests can enter search terms which will be matched in accordance with ISSN.
  - ⟨FM250R⟩ Users and guests can enter search terms which will be matched in accordance with ISNumber.
  - ⟨FM250S⟩ Users and guests can enter search terms which will be matched in accordance with metadata.
  - ⟨FM250T⟩ Users and guests can enter search terms which will be matched in accordance with abstract.
- ⟨FM260⟩ Users and guests can use an advanced search, which restricts the search to various combinations of parameters.
  - ⟨FM260A⟩ Users and guests can restrict the search to a specific research field.
  - ⟨FM260B⟩ Users and guests can restrict the search to a specific author.
  - ⟨FM260C⟩ Users and guests can restrict the search to a specific publisher.
  - ⟨FM260D⟩ Users and guests can restrict the search to a specific publication title.
  - ⟨FM260E⟩ Users and guests can restrict the search to a specific publication year.
  - ⟨FM260F⟩ Users and guests can restrict the search to a specific affiliation.
  - ⟨FM260G⟩ Users and guests can restrict the search to a specific abstract.
- 3. Users and guests can see how many times a publication or an author has been cited.
  - ⟨MR30⟩
  - ⟨FM310⟩ Users and guests can see how many times a publication has been cited excluding self-citations.
  - ⟨FM320⟩ Users and guests can see how many times an author has been cited excluding self-citations.
- 4. ScholariZer offers a Graphical User Interface (GUI) ⟨MR40⟩
  - ⟨FM410⟩ The GUI offers a home page.
    - ⟨FM410A⟩ The GUI offers a search bar to enter free text input.
      - ⟨FM410A1⟩ The GUI offers a drop down menu attached to the search bar for advanced search.
      - ⟨FM410A2⟩ The GUI offers a button attached to the search bar to execute search terms.
    - ⟨FM410B⟩ The GUI offers a button for user login.

- ⟨FM420⟩ The GUI offers a profile page, showing information and publications of a researcher
  - ⟨FM420A⟩ The GUI offers a Button to follow a researcher.
    - ⟨FM420B⟩ The GUI offers a Button to compare profiles.
    - ⟨FM420b1⟩ The GUI offers a graphical representation of the compared data about researchers.
  - ⟨FM430⟩ The GUI offers a page for search results.
- 5. Scholarizer differentiates between regular-citations and self-citations. ⟨MR50⟩
  - ⟨FM510⟩ The web app differentiates between regular-citations and self-citations in relation to h-index.
  - ⟨FM520⟩ The web app differentiates between regular-citations and self-citations in relation to i10-index.
  - ⟨FM530⟩ The web app differentiates between regular-citations and self-citations in relation to citation count.
- 6. Users and guests are able to compare multiple authors ⟨MR60⟩
  - ⟨FM610⟩ Users and guests can compare a minimum of two, up to four authors at once.
  - ⟨FM620⟩ Authors can be compared based on their h-index.
  - ⟨FM630⟩ Authors can be compared based on their i10-index.
  - ⟨FM640⟩ Authors can be compared based on their number of publications.
    - ⟨FM640A⟩ The number of publications can be filtered by year.
  - ⟨FM650⟩ Authors can be compared based on their number of citations, including and excluding self-citation.
    - ⟨FM650A⟩ The number of citations can be filtered by year.
- 7. Scholarizer offers a recommended page to its users. ⟨MR70⟩
  - ⟨FM710⟩ Scholarizer offers a recommended page based on information given in user profile followed authors.
  - ⟨FM720⟩ User specific recommendations will be hidden if not logged into user account.
- 8. Each publication is shown with a link redirecting to the web page where the publication can be read. ⟨MR80⟩
- 9. Frequent co-authors are shown in a researcher's profile. ⟨MR90⟩

10. A tab is shown on an Author's profile displaying their peers who cited them the most.  $\langle \text{MR100} \rangle$
11. The h-index and number of citations of a researcher are displayed with and without self-citations on their profile.  $\langle \text{MR110} \rangle$

## 6.2 Optional

1. Users and guests should be able to export references in plain text or BibTeX format.  $\langle \text{OR10} \rangle$
2. Users should be able to bookmark publications.  $\langle \text{OR20} \rangle$ 
  - $\langle \text{FO210} \rangle$  Users should have a quick access to their list of bookmarks.
  - $\langle \text{FO220} \rangle$  The bookmark function will be hidden if not logged into user account.
  - $\langle \text{FO230} \rangle$  Users should be able to delete a bookmark.
3. Users should be able to follow researchers.  $\langle \text{OR30} \rangle$ 
  - $\langle \text{FO310} \rangle$  Users should be able to have a quick access to a list of their followed authors.
  - $\langle \text{FO320} \rangle$  The follow function will be hidden if not logged into user account.
  - $\langle \text{FO330} \rangle$  Users should be able to unfollow.
4. There should be an option to exclude co-authors from  $\langle \text{MR100} \rangle$ .  $\langle \text{OR40} \rangle$
5. Users should get a list of recommended authors depending on their interests.  $\langle \text{OR50} \rangle$ 
  - $\langle \text{FO510} \rangle$  Recommended authors are sorted by h-index with and without self citations.
  - $\langle \text{FO520} \rangle$  Recommended authors are sorted by total citations with and without self citations.
  - $\langle \text{FO530} \rangle$  Recommended authors are sorted by i10-index with and without self citations.
6. The relative impact score of a publication or researcher should be displayed.  $\langle \text{OR60} \rangle$

7. Other optional functional requirements:

- ⟨FO710⟩ Users and guests should be shown a preview of the abstract of a searched publication.
- ⟨FO720⟩ The profile of researchers should show a link to their website if available.
- ⟨FO730⟩ Additionally to regular-citation and self-citation, ScholariZer should differentiate co-author citations in relation to indices
  - ⟨FO730A⟩ ScholariZer should differentiate co-author citations in relation to h-index.
  - ⟨FO730B⟩ ScholariZer should differentiate co-author citations in relation citation count.
  - ⟨FO730C⟩ ScholariZer should differentiate co-author citations in relation to i10-index.
- ⟨FO740⟩ Users and guests should get a preview on the keywords of a searched publication.
- ⟨FO750⟩ Scholarizer should search all publications available in the Semantic Scholar API.
- ⟨FO760⟩ ScholariZer should search all publications available in the ACM Digital Library API.
- ⟨FO770⟩ ScholariZer should offer automated registration to the user profile.
  - ⟨FO760A⟩ ScholariZer offers automated registration using a Google-account.
  - ⟨FO760B⟩ ScholariZer offers automated registration using a kit account.
- ⟨FO780⟩ For each publication there should be a graphical representation of a citation network.
- ⟨FO790⟩ There should be an option to switch between a dark and light theme.
- ⟨FO7100⟩ There should be an option to add more metrics.

## 7 Non-functional requirements

### 7.1 Mandatory

#### 7.1.1 Usability

⟨*NFUM10*⟩ ScholariZer GUI will be easy to use.

⟨*NFUM20*⟩ The GUI will be in English.

⟨*NFUM30*⟩ ScholariZer provides a user manual.

⟨*NFUM30A*⟩ The scholariZer user manual will provide an installation guide.

⟨*NFUM30B*⟩ The scholariZer user manual will describe HowTo navigate the user interface.

#### 7.1.2 Security

⟨*NFSM10*⟩ All user passwords will be encrypted.

#### 7.1.3 Performance

⟨*NFPM10*⟩ ScholariZer will have low resource consumption.<sup>6</sup>

⟨*NFPM20*⟩ ScholariZer will display search results after a maximum of one second.<sup>7</sup>

#### 7.1.4 Maintainability

⟨*NFMM10*⟩ A modular design will be applied.

⟨*NFMM20*⟩ A new Java version will be used.<sup>8</sup>

⟨*NFMM30*⟩ Apache maven will be used.

⟨*NFMM40*⟩ Full documentation of the code will be available for future development.

#### 7.1.5 Scalability

⟨*NFSM10*⟩ A modular design will be applied.

#### 7.1.6 Data

⟨*NFDM10*⟩ All needed data about publications will be stored.

⟨*NFDM20*⟩ All needed data about users will be stored.

---

<sup>6</sup>partially ensured by using React

<sup>7</sup>provided recommended product environment standards are met

<sup>8</sup>OpenJDK 19 or later, see development enviroment

## **7.2 Optional**

### **7.2.1 Usability**

$\langle NFUO10 \rangle$  ScholariZer should be mobile responsive.

### **7.2.2 Security**

$\langle NFSO10 \rangle$  After registration a verification E-Mail should be sent containing a six digit verification code.

$\langle NFSO20 \rangle$  After logout there should be a popup confirming that the logout was successful.

### **7.2.3 Scalability**

$\langle NFSO10 \rangle$  A plugin interface should be provided.

## 8 Test cases

⟨TC10⟩ Guests can register and set up a profile.

- Preconditions: The guest getting registered has no account.
- Steps:
  1. Guest clicks on the “Log in/Register” button on the website.
  2. Guest clicks on “Register now”.
  3. Guest has to type in his e-mail address and choose a password.
  4. Guest clicks on “Register”.
- Expected result: The Guest should get displayed that he got send a verification-email to activate his account.

⟨TC20⟩ Users and guests can search for publications and authors.

- Preconditions: The user/guest is on the scholariZer landing page.
- Steps:
  1. User/guest clicks onto the searchbar.
  2. User/guest types in an author.
  3. User/guest gets search results.
- Expected result: The user/guest should see the author’s profile on the top and underneath his publications.

⟨TC30⟩ Users and guests can search for publications and authors.

- Preconditions: The user/guest is on the scholariZer landing page.
- Steps:
  1. User/guest clicks onto the search bar.
  2. User/guest types in a search term.
  3. User/guest gets search results.
- Expected result: The user/guest should see the publications containing his search term.

⟨TC40⟩ Users and guests can see how many times an author has been cited.

- Preconditions: The user/guest is on the authors profile.
- Steps:
  1. User/guest sees the number of citations on top of the profile.

- Expected result: The user/guest should see correct number of times the author was cited.
- ⟨TC50⟩ Users and guests can see how many times a publication has been cited.
- Preconditions: The user/guest has searched for a specific publication.
  - Steps:
    1. User/guest sees the number of citations of the scholar in the preview.
  - Expected result: The user/guest should see correct number of times the publication was cited.
- ⟨TC60⟩ ScholariZer offers a Graphical User Interface (GUI).
- Preconditions: The user/guest is on the scholariZer.com landing page.
  - Steps:
    1. User/guest can enter a search term into the displayed search bar.
    2. User/guest can click on the displayed search-button.
  - Expected result: The user/guest should see his entered search term and should be redirected to the search results after clicking the search-button.
- ⟨TC70⟩ The web app differs between regular citations and self-citations on the profile of authors.
- Preconditions: The user/guest is on an authors profile.
  - Steps:
    1. User/guest can see the overall number of citations.
    2. User/guest can also see the number of citations without self-citation.
  - Expected result: The user/guest should see the total number of citations and the number of citations without self-citation.
- ⟨TC80⟩ The web app differs between regular citations and self-citations on the preview of publications.
- Preconditions: The user/guest has search results and looks at the preview of one publication.
  - Steps:
    1. User/guest can see the overall number of citations.
    2. User/guest can also see the number of citations without self-citation.



- Expected result: The user/guest should see the total number of citations and the number of citations without self-citation.

⟨TC90⟩ Users and guests are able to compare multiple authors.

- Preconditions: The user/guest is on an authors profile.
- Steps:
  1. User/guest can click on the compare-button.
  2. User/guest can now search and choose up to 3 authors (to compare up to 4 in total).
  3. User/guest can click on the compare-button.

- Expected result: The user/guest should see the comparison displayed as a graph of different metrics.

⟨TC100⟩ Users get shown a recommended page based on their field of expertise.

- Preconditions: The user is logged in and on the home page.
- Steps:
  1. User can click the tab “For you recommended” on the top.
  2. User gets some recommended publications displayed.

- Expected result: The user should be redirected to the page where all the recommended pages will be displayed.

⟨TC110⟩ Each publication is shown with a link redirecting to the web page where the publication can be read.

- Preconditions: The user/guest gets some publications displayed.
- Steps:
  1. User/guest searches for some publications.
  2. User/guest can click on one publication.
  3. User/guest gets redirected.

- Expected result: The user/guest should be redirected to the web page where the publications is published and should be able to read it.

⟨TC120⟩ Frequent co-authors are shown in a researcher’s profile.

- Preconditions: The user/guest is on an authors profile.
- Steps:

1. User/guest should search for an author and go on his profile.
2. User/guest can see an area with frequent co-authors.

- Expected result: The user/guest should see the an area within the profile of an author where frequent co-authors are displayed.

⟨TC130⟩ Frequent citations are shown in a researcher's profile.

- Preconditions: The user/guest is on an authors profile.

- Steps:

1. User/guest should search for an author and go on his profile.
2. User/guest should see an area where frequent citations are displayed.

- Expected result: The user/guest should see in which publications the author got frequently cited.

⟨TC140⟩ The h-index of a publication is displayed with self-citations.

- Preconditions: The user/guest searched for publications.

- Steps:

1. User/guest should search for some publications.
2. User/guest should see in the preview of the publications the standard h-index.

- Expected result: The user/guest should see in the preview of the publication the correct h-index.

⟨TC150⟩ The h-index of a publication is displayed without self-citations.

- Preconditions: The user/guest searched for publications.

- Steps:

1. User/guest should search for some publications.
2. User/guest should see in the preview of the h-index without self-citations.

- Expected result: The user/guest should see in the preview of the publication the correct h-index without self-citations.

⟨TC160⟩ User should be able to bookmark publications.

- Preconditions: The user is logged in and searched for publications.

- Steps:

1. User has to log in into his account.
2. User should search for some publications.
3. User should be able to press on the bookmark-button on the publication he wants so save.

- Expected result: The user should have the publication he wanted in his bookmark-list.

⟨TC170⟩ There should be a checkbox to exclude frequent co-authors from the list of frequent citations.

- Preconditions: The user/guest is on the profile of an author.
- Steps:
  1. User/guest should go on a profile of an author.
  2. User/guest should see the frequent citation area on the right.
  3. User/guest should be able to click on the “hide frequents co-authors”-button.

- Expected result: The user/guest should see after clicking the “hide frequent co-authors”-button the list of frequent citations without the citations of the frequent co-authors.

⟨TC180⟩ Users should be able to follow researchers to get alerts for new publications.

- Preconditions: The user is logged in and searched for an author.
- Steps:
  1. User has to log in into his account.
  2. User should search for one author.
  3. User should be able to click on the “follow”-button on the profile of the author.
- Expected result: The user should have the author in his “follow”-list and get alerts whenever the author uploads something new.

⟨TC190⟩ The relative impact score of a researcher should be displayed.

- Preconditions: The user/guest has searched for an author.
- Steps:
  1. User/guest has entered into the search bar an authors name.
  2. User/guest should get displayed on top the profile of the author.
  3. User/guest should be able to see the impact score of the author in the pre-view.

- Expected result: The user/guest should see the impact score of the author.

⟨TC200⟩ The relative impact score of a publication should be displayed.

- Preconditions: The user/guest has searched for a publication.
- Steps:
  1. User/guest has entered into the search bar a publication title.
  2. User/guest should get displayed on top the wanted publication.
  3. User/guest should be able to see the impact score of the publication in the preview.
- Expected result: The user/guest should see the impact score of the publication.

⟨TC210⟩ The profile of researchers should show a link to their website if available.

- Preconditions: The user/guest has searched for an author and went on his profile.
- Steps:
  1. User/guest has entered into the search bar an authors name.
  2. User/guest should get displayed on top the profile of the author.
  3. User/guest should be able to click on the authors name to get on his profile.
  4. User/guest should see on the top right an icon on which he can click and gets redirected to the author's website if available.
- Expected result: The user/guest should be redirected to the author's website if available.

## 9 Scenarios

### 9.1 Scenarios

#### ⟨S10⟩ *Scenario: Alex needs paper*

Alex is a computer science student. At the moment he is working on his proseminar. This means he has to collect information about current scientific work. He is supposed to sum it up to about 6 pages and talk about it in a presentation. His topic is partial configuration of field-programmable gate arrays. Alex thought the topic could be interesting, but unfortunately he has never heard about FPGAs before. In order to find out about the topic, he reads a paper recommended by his supervisor. For deeper understanding, Alex wants to search for other publications with a close relation to the topic.

To do so, he opens his Firefox web browser on his Windows PC and opens scholariZer. The website opens.

A search bar is displayed at the center of the screen. He can see a navigation bar located on the right side of the screen. The navigation bar contains a log in icon.

Alex clicks on the Login and register button which is located in the top right corner of the screen. He enters his email and password into the dialog, displayed on a popup window. An email is sent to his email address. After clicking on the validation link, Alex gets a message telling him that he successfully signed up. He is prompted to connect his university account to his scholarizer account for simple access to publications. He refuses for the moment. Alex is asked to add information to his user profile. He declines this as well. Now he finally wants to get going. To filter for specific scientific professions, he clicks on "Advanced Search" next to the search bar. Alex picks a topic from the now appeared dropdown, types in "computer science" and selects it. He also selects "electrical engineering". Next, Alex types in the name of the paper, recommended by his supervisor, and hits enter. The first result is the paper just referred to.

Each result to the search is represented by a window, showing a publications title, authors, citations and its abstract. Alex clicks on the paper. He can choose between getting access to the paper, listing related publications, listing cited publications and showing a graph representation of similar publications. Related and cited publications can be sorted by relevance, similarity, author alphabetically, same author and submission date. Alex chooses to list the publications sorted by relevance. The 20 most cited publications, related to the original one, get listed. Alex could swap to the next page to see less cited publications, but chooses not to.

Alex clicks onto the most cited publication. He is directed to [ieeexplore.ieee.org](http://ieeexplore.ieee.org). He has to log in with his university account to get access to the publication.

At this point, he can view the publication or download it. Alex reads the publication and chooses to save it, by adding it to his favorites at scholariZer.

He clicks on a button on the top right to log out from scholariZer and closes his web browser.

⟨S20⟩ *Scenario: Holger needs a new Software Engineering professor*

Due to recent job opportunities the former Software Engineering professor at KIT now works at another university. The Human Resources department of KIT now need to decide on who to pick for this job. Thus Holger will have to figure out what researcher is best suited for this position.'

Holger has heard about scholariZer, but doesn't know what it can be used for or how it is used. Holger now opens scholariZer from their work computer running Ubuntu, using Google Chrome. Holger is now welcome by a simple looking web-page containing a big search bar. Without even thinking about it too long he enters the name of the first applicant in the search bar.

He now sees a page containing among other results, the person he searched for. Clicking on this author he quickly notices by looking at the tags of the applicant, that this person isn't even a Software Engineering researcher and therefore isn't fit for the job.

He now clicks into the search bar on the top of the screen and enters the name of the second applicant. After once again clicking on the Profile of this applicant he this time sees, that they could be a possible candidate.

He now clicks on a button saying "Compare To". He is redirected to a page, where another author can be added to compare to the other, first applicant. He clicks on the plus, a search is now displayed over the plus. There he enters the third applicants name and adds them to the comparison.

Here he sees, that the third and last applicant seems to have a higher h-Index, than the second one. Now he remembers his colleagues talking about scholariZers feature to exclude self-citations from statistic calculations. He clicks on three dots on the top right of the page. He checks, that he doesn't want to see self-citations anymore and returns to the comparison screen. Here he now notices, that the second applicant actually has a way better h-Index than the third applicant.

Thanks to this knowledge Holger now knows who to employ.

## 9.2 GUI mockups



Search



Advanced Search

Figure 2: main page

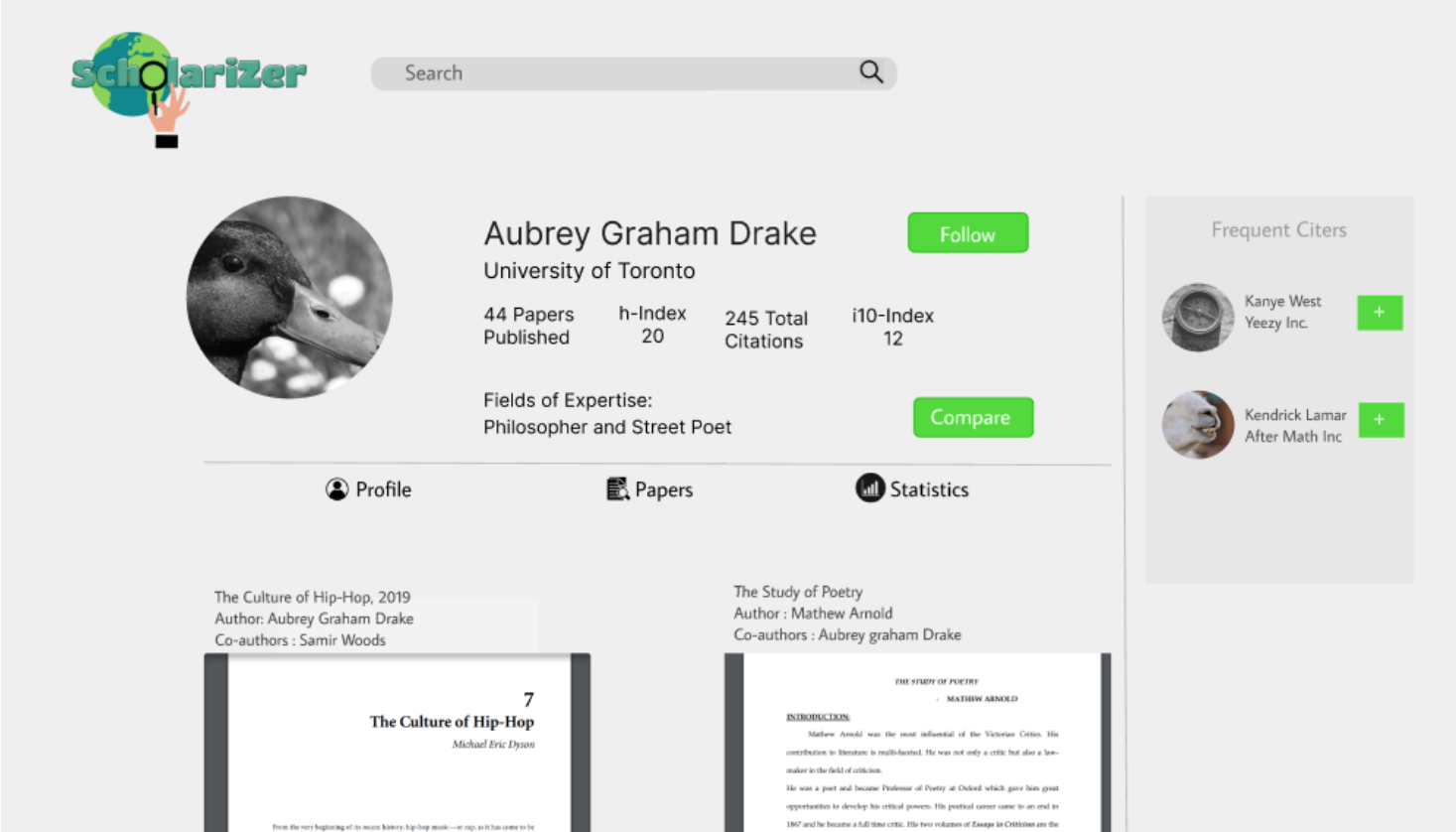


Figure 3: profile page





Search



Aubrey Graham Drake

University of Toronto

44 Papers  
Published

h-Index  
20

245 Total  
Citations

i10-Index  
12

Fields of Expertise:  
Philosopher and Street Poet

Follow

Compare

Frequent Citers



Kanye West  
Yeezy Inc.



Kendrick Lamar  
After Math Inc

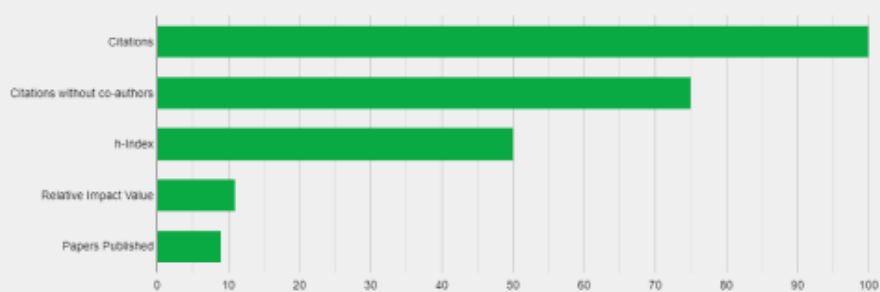


Profile

Papers

Statistics

Statistics



Papers Published Along The Years

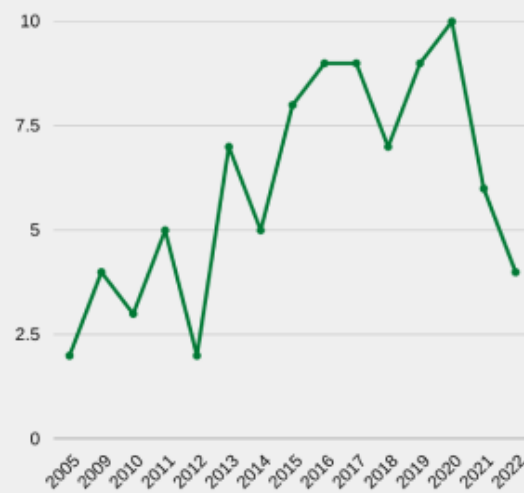


Figure 4: statistics page of a profile



Search



Showing 34 Search Results for Aubrey Graham Drake

Filters



## The Culture of Hip-Hop, 2019

Author: Aubrey Graham Drake, Co-authors : Samir Woods

... animate the Wheelchair Drake meme and consider the ways that this memetic clustersubjects Aubrey Graham to the strictures of ableist hegemonic masculinity. Aubrey Graham, plus ...

[Save](#) [Cite](#) [Related articles](#)



## Street Poetry And The Art Of Rhyming, 2014

Author: Aubrey Graham Drake, Co-authors : Samir Woods

... animate the Wheelchair Drake meme and consider the ways that this memetic clustersubjects Aubrey Graham to the strictures of ableist hegemonic masculinity. Aubrey Graham, plus ...

[Save](#) [Cite](#) [Related articles](#)



## The History of Canadian Slang, 2007

Author: Aubrey Graham Drake, Co-authors : Samir Woods

... animate the Wheelchair Drake meme and consider the ways that this memetic clustersubjects Aubrey Graham to the strictures of ableist hegemonic masculinity. Aubrey Graham, plus ...



Figure 5: freetext search



Showing 34 Search Results for Aubrey Graham Drake

Filters



- |   |   |                          |
|---|---|--------------------------|
| <input type="checkbox"/> Desc Citations | <input type="checkbox"/> Desc Publication | <input type="checkbox"/> |
| <input type="checkbox"/> Asc Citations  | <input type="checkbox"/> Asc Publication  | <input type="checkbox"/> |
| <input type="checkbox"/> Asc Title      | <input type="checkbox"/>                  | <input type="checkbox"/> |

## The Culture of Hip-Hop, 2019



Author: Aubrey Graham Drake, Co-authors : Samir Woods

... animate the Wheelchair Drake meme and consider the ways that this memetic clustersubjects Aubrey Graham to the strictures of ableist hegemonic masculinity. Aubrey Graham, plus ...

[Save](#) [Cite](#) [Related articles](#)

---

## Street Poetry And The Art Of Rhyming, 2014



Author: Aubrey Graham Drake, Co-authors : Samir Woods

... animate the Wheelchair Drake meme and consider the ways that this memetic clustersubjects Aubrey Graham to the strictures of ableist hegemonic masculinity. Aubrey Graham, plus ...

[Save](#) [Cite](#) [Related articles](#)

---

## The History of Canadian Slang, 2007



Author: Aubrey Graham Drake, Co-authors : Samir Woods

... animate the Wheelchair Drake meme and consider the ways that

Figure 6: freetext search with filters



Search



Showing 34 Search Results for Aubrey Graham Drake



**Aubrey Graham Drake**

University of Toronto

Follow

44 Papers  
Published

h-Index  
20

245 Total  
Citations

i10-Index  
12

### The Culture of Hip-Hop, 2019



Author: Aubrey Graham Drake, Co-authors : Samir Woods

... animate the Wheelchair Drake meme and consider the ways that this memetic clustersubjects Aubrey Graham to the strictures of ableist hegemonic masculinity. Aubrey Graham, plus ...

[Save](#) [Cite](#) [Related articles](#)

### Street Poetry And The Art Of Rhyming, 2014



Author: Aubrey Graham Drake, Co-authors : Samir Woods

... animate the Wheelchair Drake meme and consider the ways that this memetic clustersubjects Aubrey Graham to the strictures of ableist hegemonic masculinity. Aubrey Graham, plus ...

Figure 7: author search

Search



[Lokesh Siddhu: Research Fellow, Computer Science and Engineering, Indian Institute of Technology Delhi](#)

” 18 h 3 g 3 i10 0

[CoMeT: An Integrated Interval Thermal Simulation Toolchain for 2D, 2.5D, and 3D Processor-Memory Systems](#)

Processing cores and the accompanying main memory working in tandem enable modern processors. Dissipating heat produced from computation remains a significant problem for processors. Therefore, the thermal management of processors continues to be an active subject of research. Most thermal management research is performed using simulations, given the challenges in measuring temperatures in real processors. Fast yet accurate interval thermal simulation toolchains remain the research tool of choice to study thermal management in processors at the system level. However, the existing toolchains focu...

” 4

[PredictNcool: Leakage aware thermal management for 3D memories using a lightweight temperature predictor](#)

Recent research on mitigating thermal problems in 3D memories has covered reactive strategies that reduce memory power consumption, and thereby, performance, when the memory temperature reaches the maximum operating limit. Such techniques could benefit from temperature prediction and avoid unnecessary invocations and state transitions of the thermal management strategy. We develop an accurate steady state temperature predictor for thermal management of 3D memories. We utilize the symmetries in the floorplan, along with other design insights, to reduce the predictor's model parameters, making it lightweigh...

” 0

Figure 8: alternative search (dark mode)

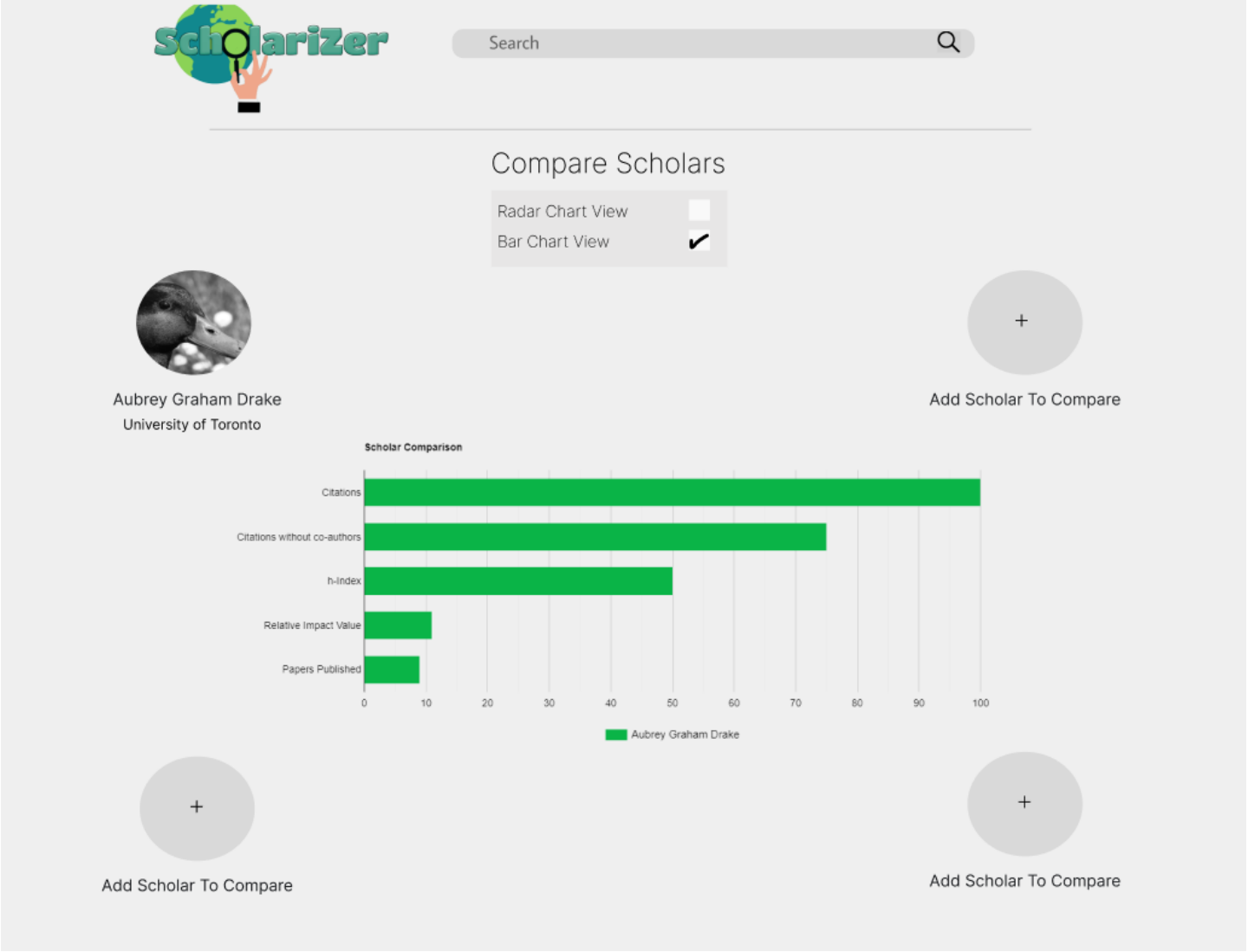


Figure 9: comparison before adding another researcher



Search



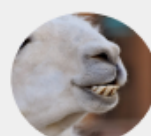
## Compare Scholars

Radar Chart View ☐

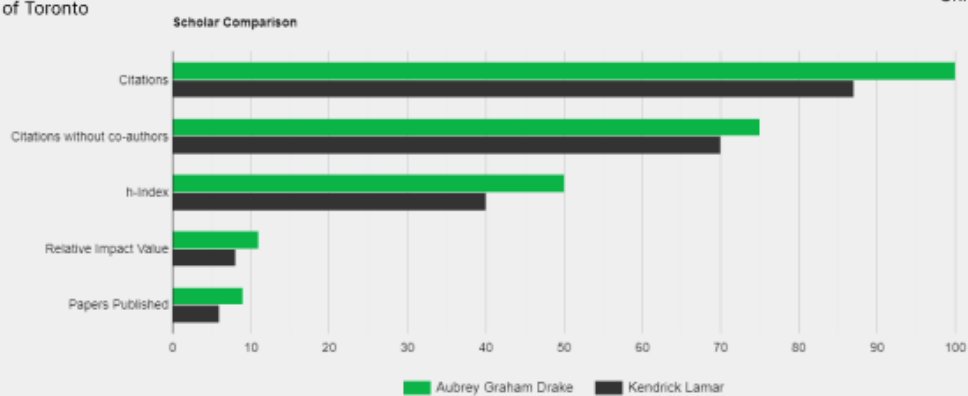
Bar Chart View ☒



Aubrey Graham Drake  
University of Toronto



Kendrick Lamar  
University of California



+

Add Scholar To Compare

+

Add Scholar To Compare

Figure 10: Comparison of two researchers



Search



## Compare Scholars

Radar Chart View ☒

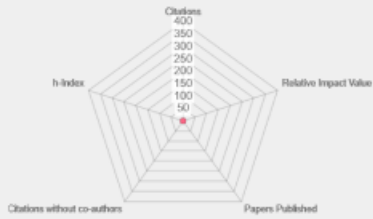
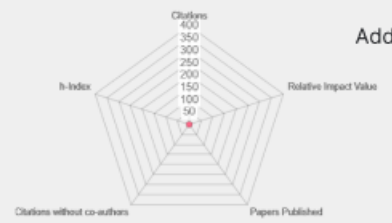
Bar Chart View ☐



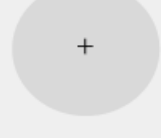
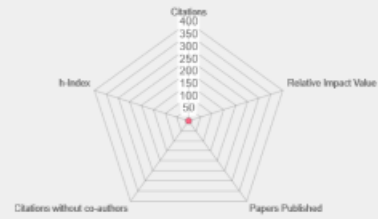
Aubrey Graham Drake  
University of Toronto



Add Scholar To Compare



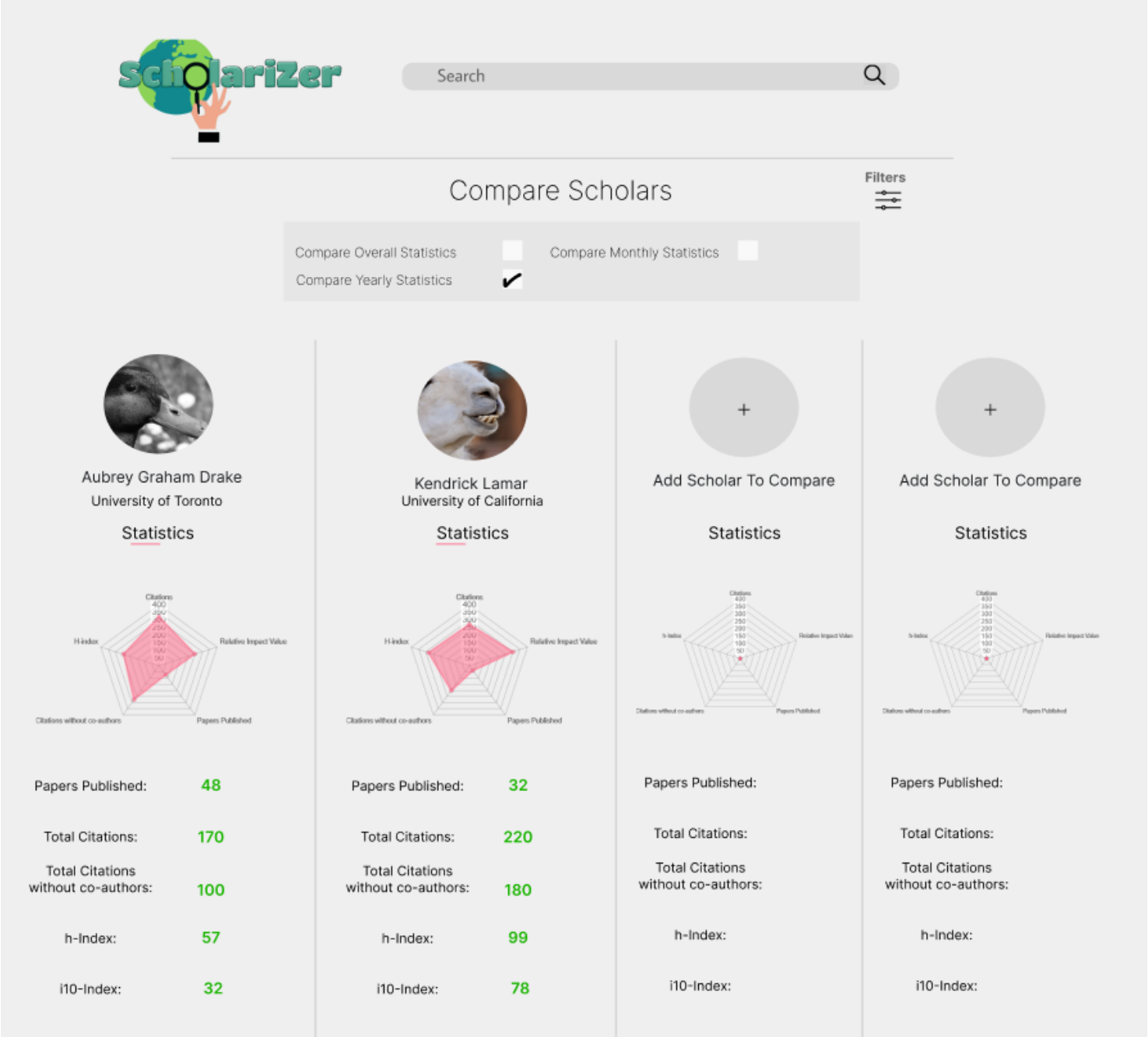
Add Scholar To Compare





Add Scholar To Compare

Figure 11: Comparison using a radar graph










---

Recommended For You



Travis Scott just published a new paper

### Astrophysics Simplified in 789 Pages, 2022

Author: Travis Scott, Co-authors : Kanye West

Astrophysics is a science that employs the methods and principles of physics and chemistry in the study of astronomical objects and phenomena,...

[Save](#) [Cite](#) [Related articles](#)




---

### The Culture of Hip-Hop, 2019

Author: Aubrey Graham Drake, Co-authors : Samir Woods

... animate the Wheelchair Drake meme and consider the ways that this memetic clustersubjects Aubrey Graham to the strictures of ableist hegemonic masculinity. Aubrey Graham, plus ...

[Save](#) [Cite](#) [Related articles](#)




---

### Street Poetry And The Art Of Rhyming, 2014

Author: Aubrey Graham Drake, Co-authors : Samir Woods

... animate the Wheelchair Drake meme and consider the ways that this memetic clustersubjects Aubrey Graham to the strictures of



Navigate

 Profile

 Search

 Explore

 Bookmarks

 Settings

Figure 13: recommended page

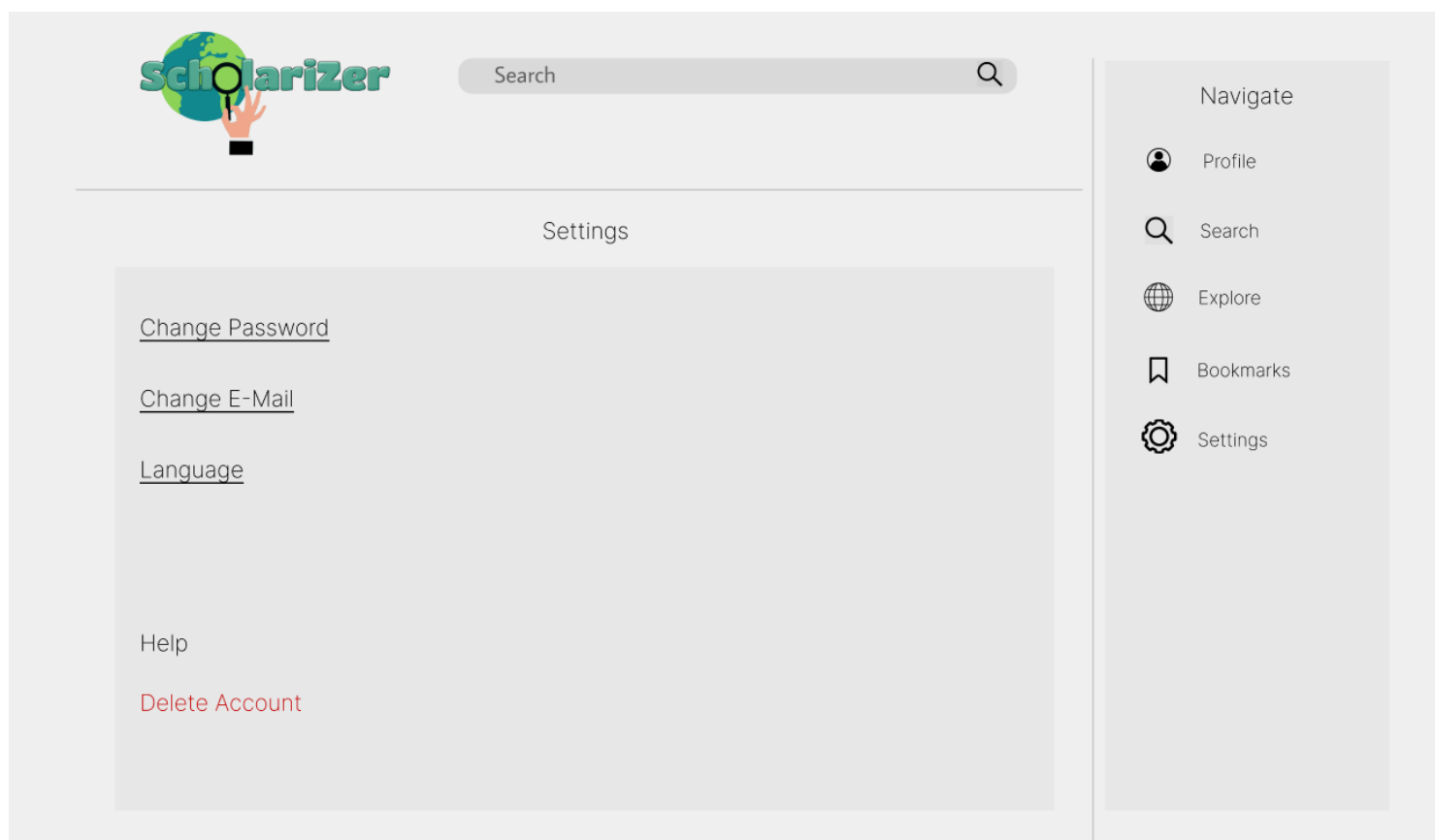


Figure 14: Settings page

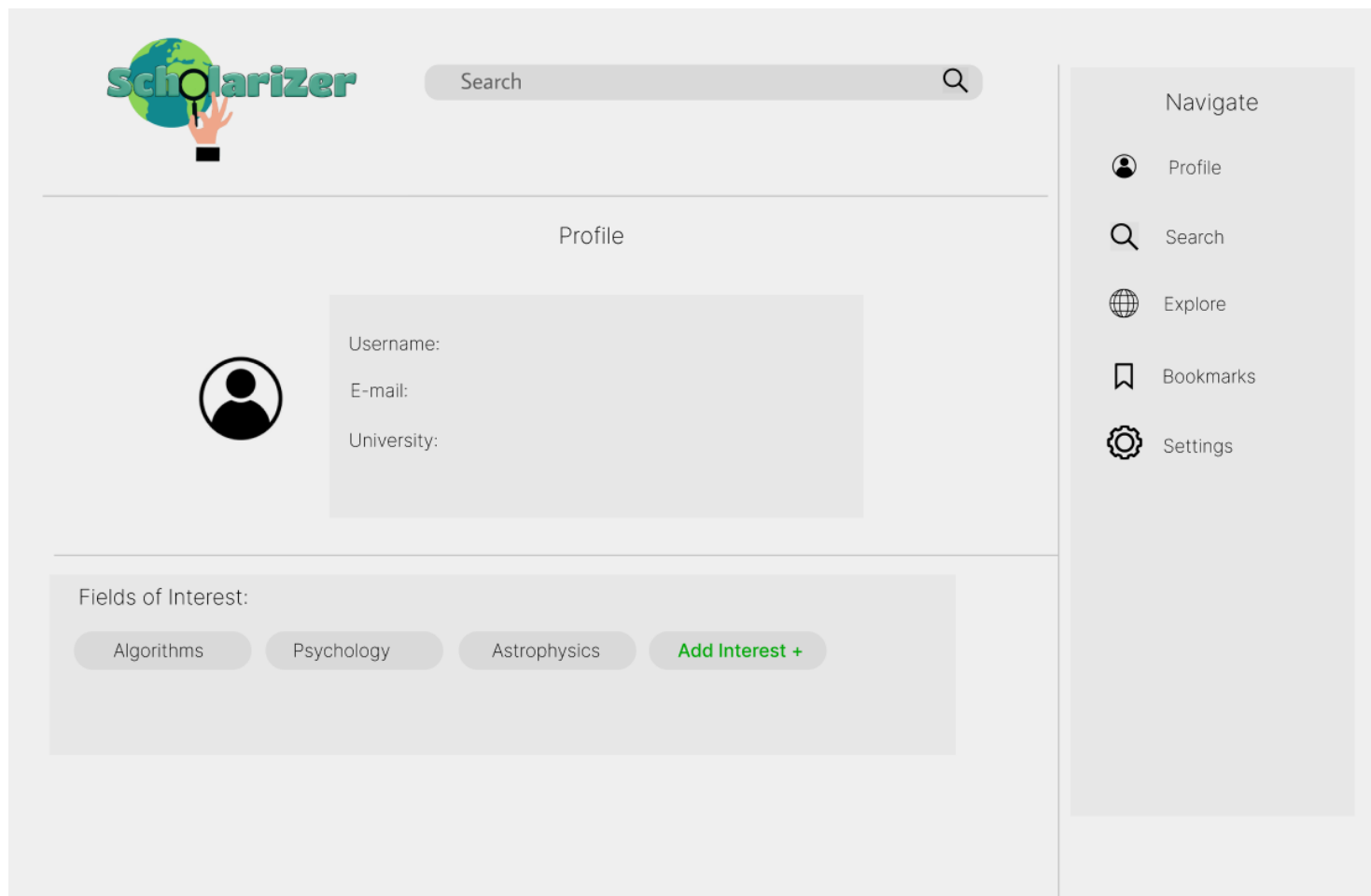


Figure 15: profile creation menu

## 10 System models

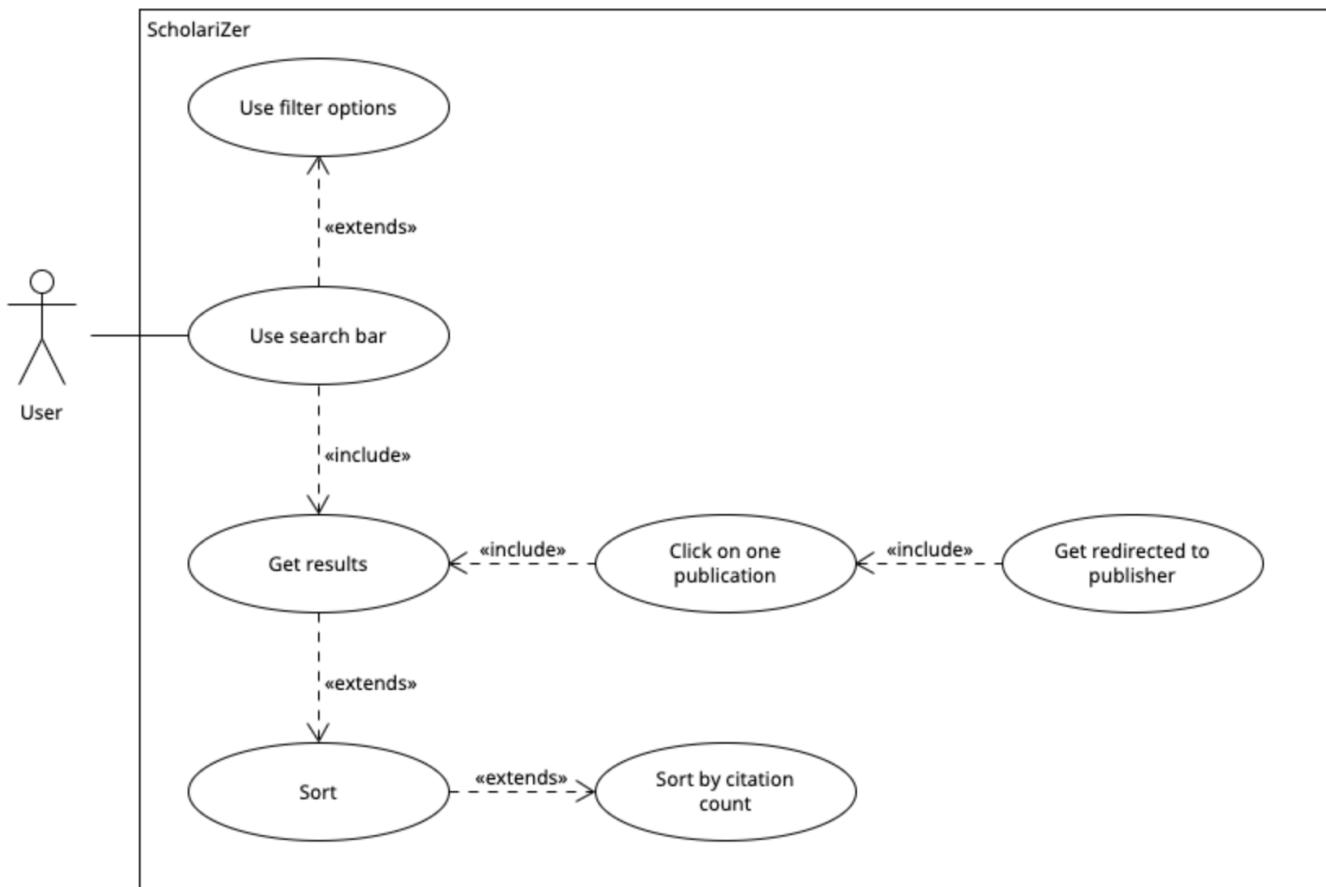


Figure 16: Search publication Use-Case

**Actors:** User

**Flow of events:**

1. User types into search bar the topic he wants to do research in
2. User can optionally already filter his search results
3. User can optionally use a sorting parameter like citation count
4. User gets all the results
5. When a result is selected, User gets redirected to the website where its published
6. User now either can view or download the publication

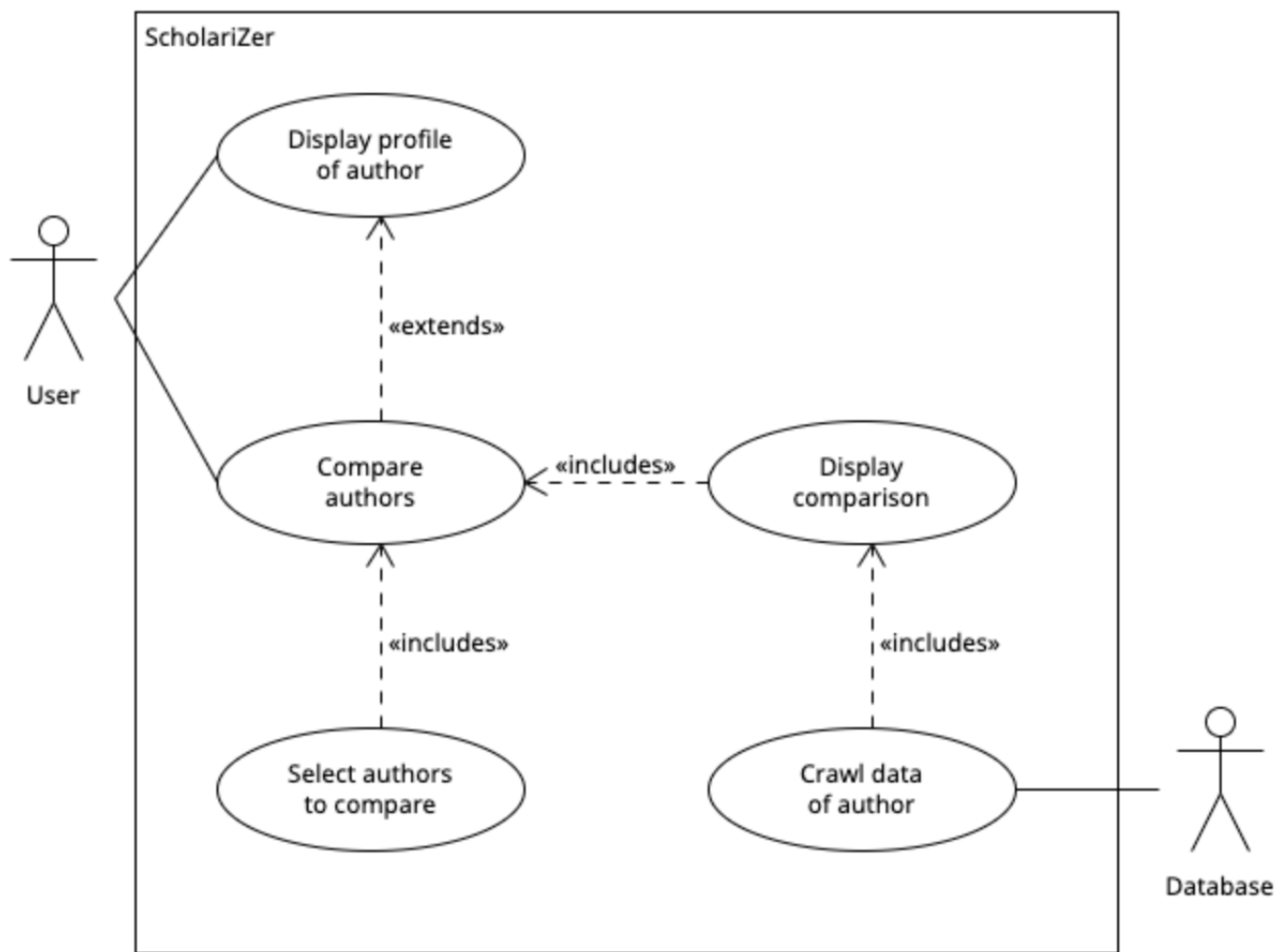


Figure 17: Compare users Use-Case

**Actors:** User, Database

**Flow of events:**

1. User has to open the profile of one author.
2. User now can click on the “compare authors” button and can add up to 3 more author he wants to compare.
3. Database is now searching for the information.
4. User gets the comparison displayed.

## 11 Development enviroment

- Database: IEEE Xplore digital library and Semantic Scholar for publications<sup>9</sup>, PostgreSQL for user profiles
- Server: Computer of KIT
- Server Operating System: A Linux OS with Docker running on top of it.
- Container management: Docker, kubernetes, Docker desktop
- IDEs: IntelliJ IDEA, WebStorm
- Frontend Language: TypeScript 4.9 with Babel as a transcompiler
- Frontend Frameworks: Next.js 13.0.5, React 18.2.0
- Backend Language: Java (OpenJDK 19) built using Apache maven 3.8
- Backend Framework: Spring using the Spring Boot Extension
- GUI design: Figma
- UML diagrams: Lucidchart, Microsoft Visio, UMLet 15.0
- Version Control: GitHub via Git, npm
- Organization and productivity tool: Notion, Google Calendar
- Code Quality inspection: SonarQube, SonarLint, ESLint, Checkstyle
- Unit Testing: JUnit, Jest
- Webbrowser to view GUI: Mozilla Firefox, Google Chrome
- Writing: Overleaf to write LaTeX
- Documentation: JavaDoc, README in Markdown

---

<sup>9</sup>data fetched through API