SMARTDORK: DESIGN OF A MODULAR SEARCH APPLICATION BASED ON GOOGLE DORK USING FLASK

Kol Arh Dr Ir Nurrahman SM, S.T., M.T., Try Haryadi Serka 202207095-E, Mayor Cpl Yohanes Dwi Cahyono, S.T.
Jurusan Teknik Rekayasa Keamanan Siber, Poltekad Kodiklat Angkatan Darat
Poltekad Kodiklatad Ksatrian Pusdik Arhanud PO BOX 52 Malang
Email: nurrudal@gmail.com, tryhary13@gmail.com, indorana2012@gmail.com

SMARTDORK: DESIGN OF A MODULAR SEARCH APPLICATION BASED ON GOOGLE DORK USING FLASK

Abstract: *In the digital era, the ability to perform efficient and targeted information retrieval has become a pivotal necessity, particularly in the fields of research, cybersecurity, and* ***Open-Source Intelligence (OSINT)****.* ***Google Dorking*** *is a powerful advanced search technique that leverages specific operators to uncover information that is not readily accessible through conventional search methods. However, the manual composition of Google Dork queries often requires a level of technical expertise, posing a significant barrier for non-technical users.*

To address this challenge, this study presents the design and implementation of ***SmartDork****, a web-based application built on the* ***Flask*** *framework. The application's core purpose is to assist users in automatically generating tailored Google Dork queries based on their specific needs. Its modular architecture is composed of five distinct search categories: general, academic journals, news articles, datasets, and people-related information.*

The application was developed using a ***software engineering approach*** *with Python programming and HTML template rendering. The system's effectiveness was validated through rigorous testing to ensure the accuracy and relevance of the generated queries based on user input for each module.*

The results demonstrate that SmartDork is capable of generating relevant and structured Google Dork queries and presenting them through an intuitive interface that simplifies complex search tasks. In conclusion, SmartDork proves to be an effective solution for users seeking to perform specific and efficient open data exploration. Future work may involve adding features for direct search execution or integration with external APIs.

Keywords: Google Dork, SmartDork, OSINT, Flask, Modular Search

Abstrak: *Di era digital, kemampuan untuk melakukan pencarian informasi yang efisien dan terarah telah menjadi kebutuhan mendasar, khususnya dalam bidang riset, keamanan siber, dan* ***Open-Source Intelligence (OSINT)****.* ***Google Dorking*** *merupakan teknik pencarian tingkat lanjut yang ampuh, yang memanfaatkan operator khusus untuk mengungkap informasi yang tidak mudah diakses melalui metode pencarian konvensional. Namun, penyusunan query Google Dork secara manual seringkali memerlukan keahlian teknis, sehingga menjadi hambatan signifikan bagi pengguna non-teknis.*

*Untuk mengatasi tantangan ini, penelitian ini menyajikan perancangan dan implementasi **SmartDork**, sebuah aplikasi berbasis web yang dibangun dengan kerangka kerja **Flask**. Tujuan utama aplikasi ini adalah untuk membantu pengguna dalam membuat query Google Dork yang disesuaikan secara otomatis berdasarkan kebutuhan spesifik mereka. Arsitektur modularnya terdiri dari lima kategori pencarian yang berbeda: umum, jurnal akademik, berita, dataset, dan informasi individu.*

*Aplikasi ini dikembangkan dengan pendekatan **rekayasa perangkat lunak** menggunakan pemrograman Python dan template rendering HTML. Efektivitas sistem divalidasi melalui pengujian ketat untuk memastikan keakuratan dan relevansi query yang dihasilkan berdasarkan masukan pengguna pada setiap modul.*

Hasilnya menunjukkan bahwa SmartDork mampu menghasilkan query Google Dork yang relevan dan terstruktur, serta menampilkannya melalui antarmuka yang intuitif yang menyederhanakan tugas pencarian yang kompleks. Sebagai kesimpulan, SmartDork terbukti menjadi solusi yang efektif bagi pengguna untuk melakukan eksplorasi data terbuka secara spesifik dan efisien. Pengembangan di masa depan dapat mencakup penambahan fitur untuk eksekusi pencarian langsung atau integrasi dengan API eksternal.

Kata kunci: Google Dork, SmartDork, OSINT, Flask, Pencarian Modular

INTRODUCTION

In today's data-driven digital era, the ability to search for information that is fast, accurate, and relevant from public sources has become an essential skill. Open-Source Intelligence (OSINT) is a rapidly growing and crucial domain across multiple disciplines, including academic research, investigative journalism, and cybersecurity (Yadav et al., 2023). While general search engines such as Google provide broad access to information, more specific or hidden data often requires advanced search techniques.

One of the most powerful and effective methods is Google Dorking, which utilizes specialized search operators (e.g., intitle:, inurl:, filetype:, site:, etc.) to uncover publicly available data that is not easily accessible through conventional searches (Abasi et al., 2020). This technique has become a fundamental tool for security professionals, journalists, and researchers to identify system vulnerabilities, track individuals, or locate overlooked information (Alabdulatif & Thilakarathne, 2025).

Despite its potential, Google Dorking is often considered complex and challenging for non-specialists. Users are required to understand specific syntax and operators, which limits its adoption among the general public (Abasi et al., 2020). A review of existing studies shows that most prior research emphasizes its application in cybersecurity contexts, particularly in vulnerability detection (Alabdulatif & Thilakarathne, 2025). However, many of these solutions take the form of manual scripts or lack user-friendly interfaces, preventing a structured and efficient search process for ordinary users (Bryushinin et al., 2022). Furthermore, systematic reviews of OSINT highlight a research gap regarding the integration of OSINT tools with intuitive interfaces or modular approaches that support customization (Browne et al., 2024).

Addressing these gaps, this study proposes the design and implementation of SmartDork, a web application intended to bridge the divide between the potential of

Google Dorking and its usability challenges. The project is guided by three main research questions:

A. How can a modular and intuitive web application architecture be designed to automate the generation of Google Dork queries?

B. How can the system help users select specific information categories (e.g., academic journals, news, or datasets) through structured search modules?

C. How can the effectiveness of the application be evaluated in terms of producing relevant queries and improving search efficiency?

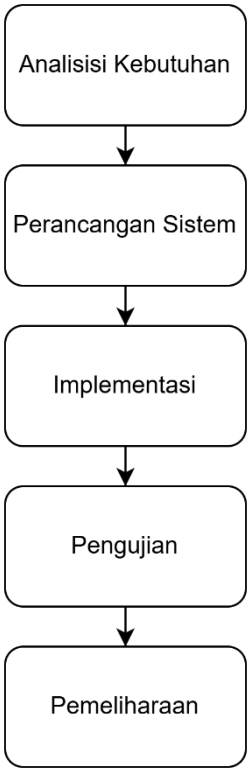
The primary objective of this research is to develop a lightweight, practical, and flexible solution using the Flask framework. By offering a simple interface, SmartDork aims to serve as an effective tool for facilitating public data exploration, enabling users from diverse backgrounds to leverage the power of Google Dorking more efficiently and with greater precision.

RESEARCH METHODOLOGY

This research adopts a software engineering approach combined with a qualitative descriptive method to design and implement a web-based application called **SmartDork**. The primary goal is to simplify advanced search techniques (*Google Dorking*) for non-technical users. The qualitative descriptive method is employed to provide a systematic and in-depth explanation of the entire development process, from requirement identification to evaluation.

The software development model applied is the **System Development Life Cycle (SDLC) Waterfall**. This model was selected due to its linear, structured, and sequential nature, ensuring that each project stage is fully completed before progressing to the next. Such an approach reduces risks for projects with clearly defined requirements and scope.

The development process follows five major stages of the Waterfall model, illustrated in the flow diagram below:



A. Requirement Analysis

This stage lays the foundation of the project, beginning with an extensive literature review to understand the landscape of Open-Source Intelligence (OSINT) and Google Dorking techniques. References such as Abasi (2020) provided insights into common use cases and adoption challenges, while systematic reviews (Browne et al., 2024) highlighted the lack of user-friendly OSINT tools.

From this analysis, both functional and non-functional requirements were identified:

- Functional requirements: The application should automatically

generate Google Dork queries, provide modular search categories (general, academic journals, news, datasets, individuals), and display queries ready for use.

- Non-functional requirements: The application must feature an intuitive and responsive interface, lightweight performance, and basic security mechanisms.

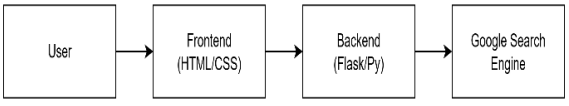
B. System Design

The identified requirements were then translated into a system blueprint, covering several aspects:

- System Architecture: SmartDork employs a simple client-server architecture. The frontend (user interface) acts as a mediator, processing requests to the backend, which generates the dork queries.

- User Interface (UI) Design: The interface prioritizes usability, ensuring forms and buttons are intuitive for users from diverse backgrounds. This aligns with the study's aim to bridge technical gaps.

- Application Logic: Each search module is designed with specific rules to combine user-provided keywords with relevant Google Dork operators.



C. Implementation

At this stage, the application source code was developed based on the system design. Technology selection was guided by project requirements and literature emphasizing lightweight and flexible web frameworks (Albesher & Alfayez, 2024).

- Backend: Implemented using Python with the Flask micro-framework. Flask was chosen for its lightweight and adaptable nature, making it well-suited for simple web applications.
- Frontend: Built using HTML for structure and Tailwind CSS for a responsive design. Jinja2 was used as the template engine to dynamically integrate backend data into the frontend.
- Search Modules: Each search category was implemented as a separate function to maintain modularity, ensuring ease of maintenance and scalability.

D. Application Testing

To verify quality, black-box testing was conducted. This method focuses on validating application functionality from the user's perspective without inspecting the internal source code. Testing scenarios included verifying outputs across different input cases. Additionally, generated queries were tested directly on Google to confirm their validity and relevance.

E. Maintenance

Although the Waterfall model is linear, the maintenance phase is treated as an iterative cycle post-deployment. It involves fixing bugs, updating search operators, and adding new features based on user feedback.

E. Data Collection and Analysis

a. Data Collection

Two main methods were applied:

- **Literature Review:** Collecting theoretical insights from journals, books, and online sources regarding Google Dorking and software development.
- **Direct Observation:** Assessing the application's functionality and performance during testing phases.

Instruments for data collection included:

- **System Logs:** Recording user inputs and generated queries to document performance.
- **User Evaluation Forms:** Gathering feedback on usability, result relevance, and improvement suggestions.

b. Data Analysis

The collected data were analyzed qualitatively through the following steps:

1. **Functionality Analysis:** Comparing SmartDork-generated queries with manual query construction to assess accuracy and effectiveness.
2. **User Feedback Analysis:** Categorizing responses from evaluation forms to identify strengths, weaknesses, and areas for improvement. These insights informed the application's future development and maintenance.

RESEARCH RESULTS

This section presents the findings from the design, implementation, and testing phases of the **SmartDork** web application. The results confirm the successful development of the system as a solution to simplify advanced search techniques, particularly Google Dorking, for users with varying technical backgrounds.

1. SmartDork Application Implementation

The SmartDork web application was fully developed following the stages of the Waterfall SDLC model. The client–server architecture proved effective in supporting smooth interactions between the frontend (user interface) and the backend (application logic).

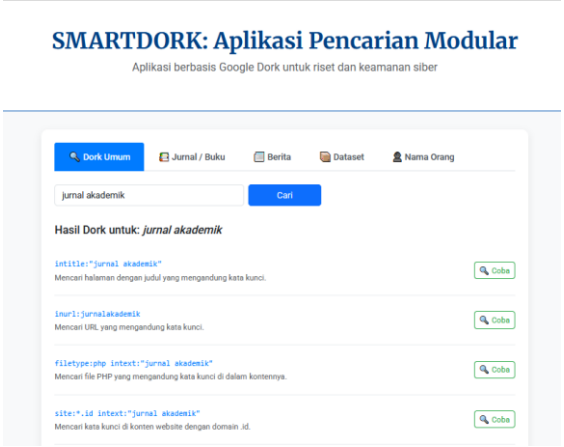
- **Backend**: The core logic was implemented in Python using the Flask micro-framework. This technology demonstrated flexibility and efficiency, aligning with findings in previous studies on Flask (Albesher & Alfayez, 2024). Five primary search modules were implemented—General, Journals, News, Datasets, and Individuals—each designed modularly. These modules process user keywords and automatically combine them with appropriate Google Dork operators, producing accurate and targeted queries.
- **Frontend**: The interface was developed with HTML and styled using Tailwind CSS, resulting in a clean, responsive, and device-friendly design. Integration with Jinja2 ensured dynamic presentation of data from the backend, creating a seamless and interactive user experience.

(Figure 1 illustrates the main interface of the SmartDork application, highlighting the five search modules.)

2. Application Functionality Testing

Functionality testing was conducted using the black-box testing method to confirm whether SmartDork generated valid and accurate queries in line with the defined specifications.

The results revealed that all modules functioned correctly and produced expected outputs. Generated queries were tested directly on Google, confirming their relevance and validating that the application successfully automated a process that is otherwise time-consuming and error-prone (Bryushinin et al., 2022).



(Figure 2 presents an example of a query produced by the “Academic Journals” module after a user entered specific keywords.)

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian	Keterangan
1	Pengguna memasukkan kata kunci di modul "Jurnal Akademik".	Aplikasi menghasilkan query dengan operator filetype:pdf dan site:edu.	Berhasil	Sintaks yang dihasilkan relevan dan akurat.
2	Pengguna memasukkan kata kunci di modul "Berita".	Aplikasi menghasilkan *query* dengan operator inurl:berita dan intitle:.	Berhasil	Sintaks yang dihasilkan dapat menemukan artikel berita terkait.
3	Pengguna memasukkan kata kunci di modul "Informasi Personal".	Pengguna memasukkan kata kunci di modul "Informasi Personal".	Berhasil	*Query* yang dihasilkan relevan.
4	Pengguna mencoba mengosongkan kolom input.	Aplikasi menampilkan pesan kesalahan atau peringatan.	Berhasil	Validasi input berjalan dengan baik.

A summary of the testing results (see Table X) indicates that all SmartDork modules consistently generated valid Google Dork syntax within their respective categories. This outcome demonstrates that the modular architecture effectively supports multiple search types without compromising accuracy.

3. Effectiveness Analysis and User Feedback

Qualitative analysis of the application’s effectiveness, supported by user evaluation data, highlights SmartDork’s role in simplifying complex search processes. The application successfully addresses one of the primary barriers identified by Abasi et al. (2020): the steep learning curve associated with mastering Google Dork syntax.

User feedback revealed high satisfaction levels in the following aspects:

- **Ease of Use:** The intuitive interface and straightforward workflow enabled users to construct advanced queries without memorizing complex operators.
- **Result Relevance:** Queries generated through SmartDork consistently delivered more specific and meaningful results compared to conventional Google searches, validating the structured approach.
- **Time Efficiency:** SmartDork significantly reduced the time needed to formulate queries, enhancing productivity for researchers, academics, and cybersecurity professionals, consistent with findings on automated OSINT data collection (Bryushinin et al., 2022).

Overall, these results demonstrate that SmartDork is not only technically functional but also effective as a practical and efficient tool for structured information retrieval.

ONCLUSION AND RECOMMENDATIONS

Conclusion

Based on the entire process of research, design, implementation, and testing, several conclusions can be drawn regarding the development of the **SmartDork** application:

1. **Successful Application Development:** SmartDork was successfully developed using the Waterfall SDLC methodology with Python–Flask for the backend and HTML/Tailwind CSS for the frontend. The modular architecture allowed the system to generate specific and accurate Google Dork queries across multiple categories, validating its contribution as a practical OSINT tool.
2. **Functional and Effective:** Results from black-box testing confirmed that all features worked as expected, consistently producing valid and relevant queries. SmartDork proved effective in simplifying advanced search techniques, making it a valuable resource for users such as academics, researchers, journalists, and cybersecurity practitioners.
3. **High User Satisfaction:** Feedback confirmed that the intuitive interface and ease of use significantly improved both efficiency and search quality. SmartDork successfully bridged the knowledge gap between complex Google Dorking syntax and practical user needs, aligning with broader objectives in OSINT research.

Recommendations

For future development, several enhancements are suggested to improve SmartDork's functionality and performance:

1. **Expanded Search Categories and Operators:** Adding more categories (e.g., social media, forums, or e-commerce) and additional operators would broaden the system's scope and usefulness.
2. **Custom Query Features:** Enabling users to create, save, and manage personalized dorks would provide greater flexibility, similar to existing dork databases (Evangelista et al., 2022).
3. **Search API Integration:** Integrating with Google Search API or alternative search engines would allow the application not only to generate queries but also to display results directly within the interface, automating the full OSINT workflow (Browne et al., 2024; Bryushinin et al., 2022).
4. **Mobile or PWA Development:** Creating a mobile version or Progressive Web App (PWA) would enhance accessibility, especially for users who frequently operate outside traditional desktop environments.

DAFTAR PUSTAKA

Abasi, R., Farooq, A., & Hakkala, A. (2020). *Google dorks: Use cases and Adaption study Reza Abasi: Google dorks: Use cases and adaption study*.

Alabdulatif, A., & Thilakarathne, N. N. (2025). Hacking Exposed: Leveraging Google Dorks, Shodan, and Censys for Cyber Attacks and the Defense Against Them. *Computers*, *14*(1). https://doi.org/10.3390/computers14010024

Albesher, L., & Alfayez, R. (2024). An Observational Study on Flask Web Framework Questions on Stack Overflow (SO). *IET Software*, *2024*(1). https://doi.org/10.1049/sfw2/1905538

Browne, T. O., Abedin, M., & Chowdhury, M. J. M. (2024). A systematic review on research utilising artificial intelligence for open source intelligence (OSINT) applications. *International Journal of Information Security*, *23*(4), 2911–2938. https://doi.org/10.1007/s10207-024-00868-2

Bryushinin, A., Dushkin, A., & Melshiyan, M. (2022). *Automation of the Information Collection Process by Osint Methods for Penetration Testing During Information Security Audit*. https://doi.org/10.1109/ElConRus54750.2022.9755812

Evangelista, J. R. G., Sassi, R. J., & Romero, M. (2022). *Google Hacking Database Attributes Enrichment and Conversion to Enable the Application of Machine Learning Techniques*. https://doi.org/10.21203/rs.3.rs-1995597/v1

Yadav, A., Kumar, A., & Singh, V. (2023). Open-source intelligence: a comprehensive review of the current state, applications and future perspectives in cyber security. *Artificial Intelligence Review*, *56*(11), 12407–12438. https://doi.org/10.1007/s10462-023-10454-y