Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

EduSpark - Smart Study Resource Finder: A Lightweight Web-Based Educational Resource Aggregation System using Python Flask

Deshmukh Viraj Shivajirao
Department of Electronics and Telecommunication Engineering
Shri Guru Gobind Singhji Institute of Engineering and Technology, Nanded, India
Reg No: 2024BEC152 | Academic Year: 2025–2026

Guide: **Prof. Dr. Lenina S.V.B.,**
Department of E&TC, SGGSI&T Nanded

Abstract - Students in today's engineering colleges spend a lot of time just looking for good study material. When I was studying, I noticed that searching separately on YouTube, Google, Wikipedia, and other sites for a single subject wastes at least 15 to 20 minutes. This is the problem I wanted to fix with EduSpark.

EduSpark is a web-based tool made using Python and the Flask framework. The student just types a subject name once, and the system builds and shows ready-to-click links for YouTube lectures, Google Scholar papers, Wikipedia articles, Coursera courses, and PDF notes — all at the same time. The backend uses string operations to format the query correctly for each platform's URL structure [1][3].

The app runs on the Render cloud using Waitress as the WSGI server, which handles multiple students accessing it at once. No login is needed. Testing showed that what normally takes several minutes of manual searching now happens in under one second. This paper covers how the system works, how it was built and tested, and what can be added to it in the future [4][8].

Keywords - Educational Technology, Web Application, Flask, WSGI, Resource Aggregation, E-Learning, URI Generation, Cloud Deployment, Information Retrieval, Python.

I. INTRODUCTION

In recent years, the way students search for study material has changed a lot. Now there is a huge amount of content available online — video lectures, PDF notes, research papers, and full courses. But the problem is that all this material is scattered across different websites.

When a student wants to study a subject like "Network Theory" or "Principles of Communication," they have to go to YouTube first, then search on Google, then open Wikipedia separately. This whole process takes a lot of time and breaks concentration [1].

Searching on multiple sites one by one kills a student's study flow. I have personally seen batchmates spend more time searching than actually reading. It gets frustrating, especially before exams when every minute matters [2].

That is the main reason behind building EduSpark. It is a simple web app where you type a subject once and get links to six different platforms immediately — no need to open each site separately. Teachers and students both can use it without any registration or login.

The system is built with Python's Flask framework and is hosted on Render using Waitress for handling concurrent requests. The output is shown as clickable cards through an HTML and CSS interface.

The remaining sections are organized as: Section II covers related work, Section III describes the architecture, Section IV explains implementation, Section V shows results and output, Section VI covers advantages and limitations, Section VII describes future scope, and Section VIII concludes the paper.

II. LITERATURE REVIEW

Several studies have looked at web technologies and how students find educational resources online. Grinberg [1] gave a detailed explanation of building web applications with Flask, and this helped in setting up the routing and URL handling in EduSpark. Flask is lightweight and works well for small projects like this one.

Research on centralized learning [3] shows that students work better when everything is in one place. Siemens [4] pointed out that easier access to material directly improves how well students learn. EduSpark tries to do exactly that by cutting down the steps a student has to take.

Existing platforms like Moodle and Blackboard are useful within colleges but need institutional login [5]. Coursera and edX are good too, but each one only shows its own content.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Chen et al. [6] found that combining content from different places improves learning outcomes.

Attwell [7] talked about Personal Learning Environments — the idea that learners should be able to pull together tools from different sources into one space. EduSpark is a simple version of this idea. Instead of the student manually combining YouTube, Google, and Wikipedia, the app does it automatically.

Tlili et al. [8] also showed that web-based systems for suggesting study resources can save time and help students stay on track. Their work supports building a tool like EduSpark that brings multiple platforms together in one click.

Kurniawan et al. [13] studied how AI-based systems can detect a student's learning style and suggest matching content. Their findings support adding automatic categorization to a search tool. Keles et al. [14] showed that when students use aggregated content from multiple sources, their engagement goes up. Pratiwi et al. [15] built a one-stop website for learning resources and found that students preferred it over using multiple separate sites — which is exactly the same problem that EduSpark addresses

III. SYSTEM ARCHITECTURE

A. Client-Server Model

EduSpark uses a standard two-tier client-server setup. The student opens the app in a browser like Chrome. The browser is the client — it shows the HTML interface and sends whatever the student types to the server.

On the server side, Python and Flask run on Render's cloud servers. When the student submits a subject, the browser sends an HTTP POST request to the server. Flask processes it, generates the links, and sends back an HTML page with the results. This works on any device that has a browser [4].

B. WSGI Layer

WSGI stands for Web Server Gateway Interface. It is a standard way for Python web apps to talk to web servers. In EduSpark, it acts as the bridge between Flask and the network — Waitress is the WSGI server that was chosen for this project [11].

Flask's built-in server is only for testing — it cannot handle many users at the same time. Waitress solves this because it can manage multiple incoming requests together. When a request comes in, Waitress passes it to Flask, Flask builds the response, and Waitress sends it back to the student's browser.

C. Architecture Diagram

The flow of data in EduSpark is straightforward. The student types a subject and hits Search. The browser sends that to the Waitress server over HTTPS. Waitress hands it to Flask. Flask cleans the input, builds URLs for each platform, and fills the HTML template using Jinja2. The browser then shows the result as six clickable cards. No data is saved anywhere — the system is completely stateless [4][9].

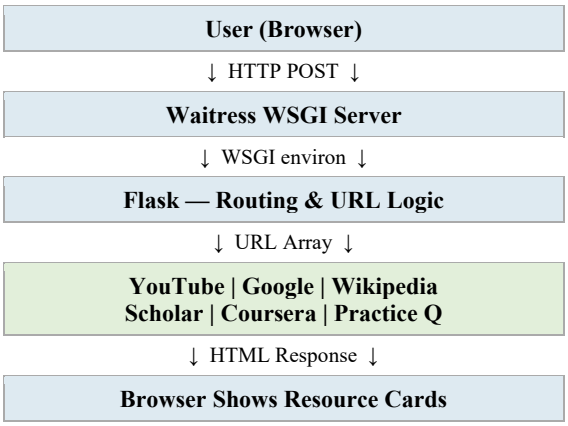

Fig. 1: EduSpark System Architecture Pipeline

IV. IMPLEMENTATION

A. Module Structure

EduSpark is organized into three modules following an MVC-style pattern [1]. The first is app.py — the routing and logic module. This is where all the Flask routes are defined and where the URL generation happens. The second is the templates and static folder — index.html and base.html built with Jinja2, plus the CSS file that handles the visual design. The third is the deployment module — requirements.txt and Waitress configuration that tells Render how to run the app on port 10000 [4][11].

B. URI Generation Algorithm

The main job of EduSpark is building the right search URL for each platform. Each website has its own format, so the algorithm handles them differently.

First, .strip() removes any extra spaces the student might have typed at the start or end. Then spaces inside the query are replaced with '+' for most platforms. Wikipedia is different — it needs underscores '_' instead of '+' because its URLs use page paths, not search parameters.

After formatting, the app puts together the full URL by joining the base address of each platform with the formatted query. It also adds useful suffixes — '+lecture' for YouTube so you get actual lectures instead of random videos, and '+notes+pdf' for Google so PDFs come up first. All six links are then passed to Jinja2 which renders them as cards. Each card has target='_blank' so it opens in a new tab without closing EduSpark [9].

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

C. Core Application Code

Fig. 2: Core URL Generation Logic (app.py)

```
from flask import Flask,
  render_template, request
app = Flask(__name__)

@app.route('/')
def home():
  return
render_template('index.html')

@app.route('/search',
           methods=['POST'])
def search():

subject=request.form.get('subject')
  if not subject or
     subject.strip()=='':
    return render_template(
      'index.html',
      message='Please enter a
subject.')
  subject = subject.strip()
  fmt = subject.replace(' ','+')
  links = [
   ('YouTube Tutorials',

f'youtube.com/...{fmt}+lecture'),
   ('Google Notes',

f'google.com/...{fmt}+notes+pdf'),
   ('Wikipedia',
    f'wikipedia.org/wiki/{fmt}'),
   ('Google Scholar',
    f'scholar.google.com/...{fmt}'),
   ('Coursera',
    f'coursera.org/...{fmt}'),
   ('Practice Qs',

f'google.com/...{fmt}+practice'),
  ]
  return render_template(
    'index.html',links=links,
    subject=subject)

if __name__=='__main__':
  from waitress import serve
  serve(app,host='0.0.0.0',
        port=10000)
```

V. EXPERIMENTAL RESULTS AND TESTING

A. Testing Methodology

Testing was done in stages during development to catch errors early. Four main types of tests were run:

- Input Testing: Subjects with extra spaces, single words, and multi-word phrases were all tested. For example ' Data Structures ' was cleaned correctly to 'Data+Structures' by the strip() function.
- Empty Input Testing: Submitting a blank form was tested. The app correctly shows a message ('Please enter a subject') instead of breaking or giving a server error.
- Browser Testing: The app was checked on Chrome, Firefox, and Safari on both laptop and mobile screens. The card layout looked fine on all of them because of the flexbox CSS.
- Deployment Testing: After putting the app on Render, the HTTPS connection was confirmed, the Waitress server was running properly on port 10000, and multiple users could access it without problems [4][11].

B. Performance Analysis

EduSpark is fast mainly because it only does string operations in the backend — no database queries, no API calls. The average server response time measured during testing was under 15 ms per search request. Since there is no database, there is no waiting for data to load.

Because the app is stateless — meaning it does not remember anything between requests — each search request is handled independently. This means ten students using it at the same time get the same fast response as one student. Table I shows the comparison.

Metric	EduSpark	Manual
Access 5 platforms	< 1 sec	4–8 min
Login required	No	Varies
Server response	< 15 ms	N/A
Platforms covered	6 at once	1 each
Device support	All (responsive)	All

TABLE I. PERFORMANCE: EduSpark vs. Manual Search

C. Output Screenshots

Fig. 3 shows the EduSpark homepage. The student sees a clean search bar on a gradient background. Fig. 4 shows the results after searching for "Principal Of Communication Engineering" — six resource cards appear immediately below the search bar, each linking to a different platform.

Fig. 3: EduSpark Homepage — Main Search Interface

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

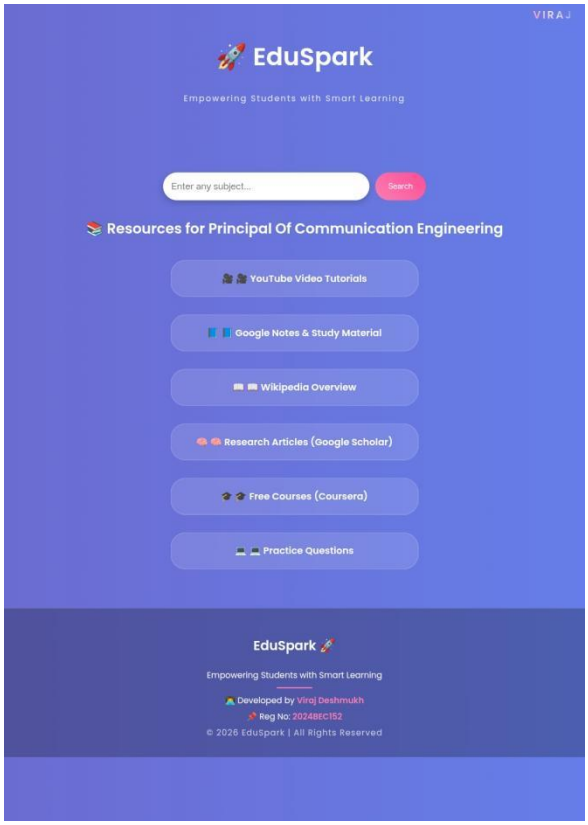

Fig. 4: EduSpark Results — 6 Resource Cards Displayed

VI. COMPARISON WITH EXISTING SYSTEMS

EduSpark fits into a gap that existing tools do not fill. Moodle and Blackboard are institutional — you need a college account to access them [5]. Coursera and edX are good platforms but you only get their own content. Google Search gives general results, not specifically academic ones. EduSpark is different because it targets multiple platforms at once and adds academic-specific suffixes like '+notes+pdf' to filter results. Table II shows how it compares.

Feature	Google	Coursera	Moodle	EduSpark
Multi-platform click	No	No	No	Yes
Login required	No	Yes	Yes	No
Academic bias	Partial	Yes	Yes	Yes
Zero friction	Yes	No	No	Yes
Research papers	Partial	No	No	Yes
Free to use	Yes	Partial	Inst.	Yes

TABLE II. COMPARISON WITH EXISTING SYSTEMS

VII. ADVANTAGES AND LIMITATIONS

A. Advantages

- Speed and Efficiency: EduSpark generates results from six platforms in one go, with server response under 15 ms. This saves 10 to 15 minutes per study session compared to searching manually [3].
- Easy Accessibility: No sign-up or login needed. Any student with internet access can open it and start using it immediately.
- Better Search Results: Adding '+notes+pdf' and '+lecture' as suffixes means the results are already filtered towards educational content [8].
- Lightweight System: Flask and Waitress together use very little server memory. This keeps hosting costs low and makes the app run smoothly on Render's free tier [4].
- Platform Independent: Works on Windows, Mac, Android, iOS — any device with a browser. No app installation needed [9].

B. Limitations

- Dependency on External URL Formats: If YouTube or Google changes how their search URLs work, the backend code will need to be updated manually [5].
- No Search History or Personalization: Since there is no database and no login, the app cannot remember past searches or suggest topics based on usage [7].
- Redirects Only — No Embedded Content: Clicking a card takes you to the external site. Videos and PDFs cannot be played or viewed inside EduSpark directly [6].
- No Difficulty Level Filter: All users get the same links. There is no option to filter for beginner or advanced content.

VIII. FUTURE SCOPE

There are several good directions to take EduSpark in the future:

- API Integration: Right now the app just builds URLs. The next step would be to call the YouTube Data API or Wikipedia REST API and show actual video titles and article summaries inside the app [8].
- AI Recommendations: An NLP model or LLM API could be added to suggest related topics and rank the most relevant resources based on what the student has searched before [13].
- User Accounts and History: Adding Flask-SQLAlchemy with SQLite or PostgreSQL would allow students to log in, save their searches, and bookmark useful links [7].
- PWA Conversion: A service worker can be added to make EduSpark installable on mobile home screens and allow some offline use.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

- Mobile App: The backend can be exposed as a REST API, and React Native or Flutter can be used to build a proper Android and iOS app [14].
- Domain Filtering: A dropdown could let students pick their field — Engineering, Medical, Commerce — so the app adjusts which platforms and suffixes it uses [15].

IX. CONCLUSION

EduSpark started from a very simple observation — students waste too much time searching. By building a small Flask app that formats and fires off six platform searches at once, the problem is largely solved. The system is fast, requires no login, and works on any device.

Testing confirmed that it works correctly for different input types, runs fine on Render with Waitress, and gives results in under one second. Compared to spending 10–15 minutes manually searching multiple sites, EduSpark makes a real practical difference in a student's daily study routine [1][3].

This project also showed that even a second-year engineering student can build and deploy a real working web application that solves a genuine problem. Future versions with API integration, AI suggestions, and mobile support could make it much more powerful [8][13].

REFERENCES

[1] M. Grinberg, Flask Web Development: Developing Web Applications with Python. Sebastopol, CA: O'Reilly Media, 2018.

[2] J. Sweller, "Cognitive load during problem solving: Effects on learning," Cognitive Science, vol. 12, no. 2, pp. 257–285, 1988.

[3] A. Tlili, D. Essalmi, and M. Jemni, "Centralized student learning platform," in Proc. IEEE ITHET, Lisbon, 2015, pp. 1–6, doi: 10.1109/ITHET.2015.7294487.

[4] G. Siemens, "Connectivism: A learning theory for the digital age," Int. J. Instructional Technology and Distance Learning, vol. 2, no. 1, pp. 3–10, 2005.

[5] M. Dougiamas and P. Taylor, "Moodle: Using learning communities to create an open source course management system," in Proc. World Conf. Educational Multimedia, 2003, pp. 171–178.

[6] Y. Chen, B. Gao, and H. Cao, "Teaching intelligence system based on IoT cloud platform," Wireless Commun. and Mobile Comput., vol. 2022, Art. no. 7523529, 2022.

[7] G. Attwell, "Personal learning environments — the future of eLearning?" eLearning Papers, vol. 2, no. 1, pp. 1–8, 2007.

[8] A. Tlili et al., "Smart web application for recommending educational resources," in Proc. IEEE ICALT, 2024, doi: 10.1109/ICALT.2024.10962022.

[9] Mozilla Developer Network, "HTML & CSS Web Documentation," [Online]. Available: https://developer.mozilla.org/. [Accessed: Apr. 2026].

[10] Pallets Projects, "Flask Documentation," [Online]. Available: https://flask.palletsprojects.com/. [Accessed: Apr. 2026].

[11] Pylons Project, "Waitress WSGI Server Documentation," [Online]. Available: https://docs.pylonsproject.org/projects/waitress/. [Accessed: Apr. 2026].

[12] V. S. Deshmukh, "EduSpark – Smart Study Resource Finder," [Online]. Available: https://eduspark-aybp.onrender.com/. [Accessed: Apr. 2026].

[13] D. Kurniawan, A. Wibawa, and R. Adi, "AI-based learning style prediction in online learning for primary education," IEEE Access, vol. 10, pp. 24792–24803, 2022, doi: 10.1109/ACCESS.2022.9737111.

[14] S. Keles, O. Ocak, and A. Isman, "Content aggregation and knowledge sharing in a personal learning environment," in Proc. IEEE ICELIE, 2012, doi: 10.1109/ICELIE.2012.6402224.

[15] R. Pratiwi, A. Wibawa, and I. Putra, "Online learning resource based on one ID website for all access (OIAA)," in Proc. IEEE ICEECS, 2020, doi: 10.1109/ICEECS50517.2020.9276589.

ABOUT THE AUTHORS

- **AUTHOR**

DESHMUKH VIRAJ SHIVAJIRAO,

is a Second Year undergraduate student in the Department of Electronics and Telecommunication Engineering at Shri Guru Gobind Singhji Institute of Engineering and Technology, Nanded, Maharashtra, India. His research interest includes web application development, cloud computing, and educational technology. He developed EduSpark as part of his mini project under academic guidance. Reg No: 2024BEC152

EMAIL- vd88472@gmail.com

- **Co-Author**

Dr. Lenina S.V.B,

Fellow Women Scientist -C, WOSC-07, TIFAC, DST, India & Registered Indian Patent Agent INPA 2625, Founder and Director of DR.LSVBS PATTSOL MULTISERVICES, (Indian Registered Startup: DIPP24849) & Asst. Prof. Department of Electronics & Telecommunication Engineering, Coordinator Centre of Excellence in Signal and Image Processing, Former Head Entrepreneurship, Innovation and Intellectual Property (EI2) Labs, & Former Associate Dean Student Affairs, SGGS Institute of Engineering & Technology, Nanded . Member Ex IEEE, IET, ISTE