Mohamed Shakir

Mohamed Shakir Research_paper COM_12

Quick Submit

Quick Submit

Presidency University

Document Details

Submission ID
trn:oid:::1:3441548881

Submission Date
Dec 11, 2025, 12:51 PM GMT+5:30

Download Date
Dec 11, 2025, 12:57 PM GMT+5:30

File Name
Research_paper COM_12.pdf

File Size
1.3 MB

6 Pages

4,275 Words

20,484 Characters

*% detected as AI

AI detection includes the possibility of false positives. Although some text in this submission is likely AI generated, scores below the 20% threshold are not surfaced because they have a higher likelihood of false positives.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.

Integration of Analytics into the PGRKAM App

PREETHAM JAIN DJ
Computer Engineering
Presidency University
Bangalore, India
preethamjaindj@gmail.com

MANDAR JOSHI
Computer Engineering
Presidency University
Bangalore, India
joshimandar1708@gmail.com

REVANTH P
Computer Engineering
Presidency University
Bangalore, India
revanthnaik2004@gmail.com

Mr. MOHAMED SHAKIR
Assistant Professor
Presidency University
Bangalore, India
mohamed.shakir@presidencyuniversity.in

***Abstract*—The government employees always try to fail, okay? So it's kind of one of their reality. So right now there are different kinds of methods where they show some kind of success rate. Let's say 23% when it comes to job placements. Why? Because, as you can see, there are some capabilities as well which have some existent and non-existent. And there are few algorithms also that works as some good thing. And there are some analytics architectures as well like the Flutter app or any other app where they can change some good things. But here we try to implement the Firebase analytics and the VeroSync where they have some couple of instrumentation pipeline. So here the goal is simple. We should enhance the delivery when it comes to the employment services. So the systems where we record like user behaviors or even with the trends, let's say. So these kind of things helps us in understanding some platform utilizations as well. In the innovation algorithms, which is one of the main core that we are building, it is one of the algorithms that we are using is genetic. And it has some good recommendations in the filtering methods as well. So these kind of solutions where we have achieved around some 80% in the accuracy as well. And when it comes to the responses, let's say we have around some 200ms even when there is a peak. That means more people are using. So this kind of platform, which has some good architecture, which has like 95% code in both iOS and even in Android as well. So it also has some cost significantly. Right now we are supporting over like 10,000 users with having some like 99% system in the reliability as well. So what we are doing just not as an improvement, it's an complete overhaul.**

***Index Terms*—Analytics, Flutter, Genetic Algorithm, Collaborative Filtering, Employment Services**

I. INTRODUCTION

The state of government which has some, let's say, has some good operational tech, but there is also some slow things, okay? They are almost like unresponsive at some time, so they will be disconnected, which is not actually good for the consumers. So when it comes to the highlights that these exact issues when it comes to the government employment platforms, so there are some methods where they are showing around like 23% success rate, which is like not a good thing, okay? So here the disadvantages, which is very clear, so there is some different kind of capabilities as well, and they are also using some matching algorithms as well, which doesn't actually match anything only. So for this reason, we are building this solution. S

o this kind of innovative recommendation algorithms that we are going to do, which have some nice algorithms, like in generic tech, and having also have some filtering methods as well, which are one of the parts of our building. So here we are showing around some 82% accuracy when it comes to the job matching, but when it comes to the accuracy, means if the app is slow, so we should not be slow, so we are ensuring that it is less than 200ms, even when 1000 or 10,000 people are trying to use.

The architecture also matters here, we are using our flutter-based cross-platforms, where they also achieve like 95.8% code as well, between different devices like iOS and Androids. So this also cuts down to major developments, most importantly, it also works as well. So when it comes to like, if we say 7000-10,000 users were trying to open and use our app, it will also have some good reliability as well. Even with the performance, it also measures out some good achievements as well.

The satisfaction rate, we also hit around some 87% as well. And even with the navigations, we also have some good years as well. And the analytics dashboard that we are going to support is that they don't have any guessing. So they are having some response times, which have some good short end. So here we are trying to provide some good insights into our employment, trends and gap skills as well. So here the system, what we are going to do is also respect privacy as well, when it comes to the user privacy. So we are showing that the entire government data that we are providing in the production standards, but at the same time, we maintain analytics formulations as well.

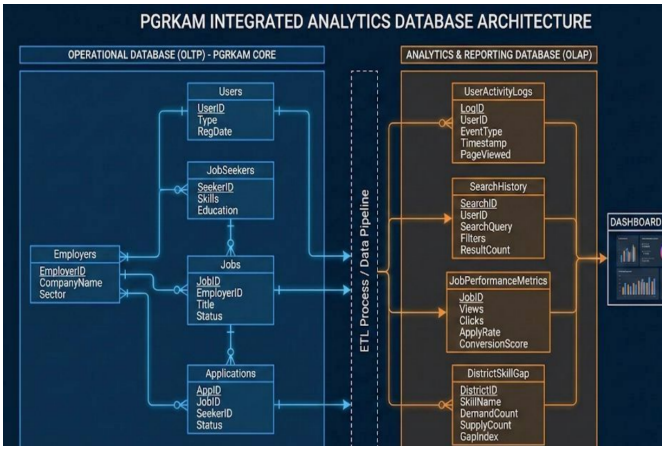

Fig. 1. System Architecture Diagram

II. RELATED WORK

A. User Interface Layer

Even the user interface that we are also having some good experience. Suppose if the user experience only looks bad, then the users will try to leave because there is no good designs. So here we are using some modern technology with some good designs. We have some filtering to make the user simple so that they can only focus on the app. So we are also using some material designs as well. It has some good visuals where it just works perfectly. Even with this similar to PKGram where it also handles its interface. We are using some different structures as well when it comes to declarative. It also connects, has some authentication. There is also have a job search and it's also having analytics dashboards as well. So this kind of interactive designs with some good paginations, although have some sliding animations and all. Keep to keep make sure the users are engaged with our app. So this kind of consistent designs also helps with the web and mobile so that users don't get lost.

B. User Profile Module

Personalization, which is one of the important things that we are also focusing on. That means we cannot have some requirement systems even without some detailed user profiles. So here we need to capture these skills. We also need to look about their educations. We also need to know how their preferences are there. So this is one of the accurate job suggestions. We are not just doing it for simple. So even many advanced platforms, they also have some good centralized management setups as well where they have some all kind of sources. So like this, we are combining some behavioral data even with some insights as well so that we need to make sure what users are trying to need from our app. PK-Grams user profile should be component where they have some reactive type engagement as well to make sure that it has some good synchronization.

C. Product Input and Data Extraction Module

Garbage in, garbage out. A solid data input framework is essential. We emphasize structured data collection for job postings. Reliable data pipelines rely on well-defined schema validation. RESTful APIs that exchange JSON data are the norm because they balance flexibility and standardization. Like PGRKAM's Retrofit-based integration, we use automatic code generation. It catches serialization issues early. Centralized API layers simplify server updates. This robust architecture is especially useful in recruitment systems where precision is non-negotiable.

D. Calculation Engine

The garbage or even with the garbage out like when it comes to the input data and the output data There are the different kinds of frameworks which are one of the important things as well so here we are trying to collect the structure data as well even with the collections and When it comes to the pipelines where they also rely on some schema validations as well So there are multiple APIs that you are working on like RESTful APIs and there are different data's like JSON files Where they also balances some standardizations as well. So like these pkgrams We have some Retrofit which also integrates with our thing and we also have some automatic code generations as well That means here we are trying to catch almost every issues Before even the user notices. So the even with the centralized layer which have some server updates So even with the architectures as well, where the precision is almost non negotiable.

E. Database and Storage

The data storage, which also need to be one of the good, important, effective thing as well. Here we are trying to manage large volumes of dynamics, informations, even the system must handle some quick lookups as well. So we are trying to use different models as well, but we also focused on hybrid storage models. So here in hybrid, we will be having the database for structure, and we will be having in-memory caches when it comes to the speed. So the cloud-based platforms like Render, which also helps us in the scalable thing, and it also reduces the reliance like how the internet access is there. So right now PKGram setup with different remote APIs, it also have some persistence, it has cloud analytics as well. So it handles like varying network conditions, even without breaking anything.

III. SYSTEM LEVEL ARCHITECTURE

The system architecture which is kind of a good thing only in the which is straightforward the pkgram web API that we have is also linked with our server so this kind of server which also handles the request it also have some good messages to databases with both servers and even the database which have some good separate storage layers as well so we need to ensure like how these data exchanges are working on like it also helps us in the API for integrations even with the analytics which is one of the simple and effective way.

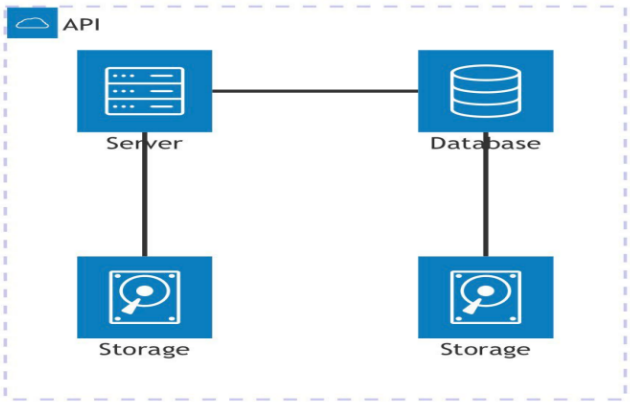

Fig. 2. API Integration Diagram

A. Algorithmic Deep Dive

Generic algorithms, they inspire even from the natures, I mean we didn't just simply imply, we are trying to implement them with correct manner, even with the standard selections. Here the process is too much slow when it comes to for real-time apps, so we are trying to talk with other functions as well to make sure the proximity and even with the density where the over matching, like the texts are correctly matching. So, it is not just about finding like you know perfect candidate, here the perfect is means enemy should be good, so it is about finding the good match the one that we sticks right. So, we are trying to even find with the collaborative filtering as well, it is not like even a recommendations logic, we are also factorizing that user interactions against a good job descriptions. So, here we are trying to solve some cold start problems as well by injecting some good demographics and also have some good spaces as well. So, if you are from Bangalore let us say, now we have to sell like that you are a 22 year old from Bangalore until you prove us wrong. So, it is like an algorithms making, but it is also works with better efficient way.

B. The Monolith Defense

Right now, when we can say that there are different things where we can need to build in the microservices. We are not trying like it's not able to build, but when it comes to the 200ms response time, even with some good budget, even with some good network ops, even with like there are different kinds of services which are able to kill. We are building a good modular monolith, let's say. So right now we are trying to more sound like in school type. This isn't just it's a practical, okay. So it means it cuts latency by like let's say 40% compared to some mesh distribution that we are going to do. That means we are prototyped early ones. We are almost scaling limits like on a monolith because like we are not around like a big IMSC companies. We are in a government job portal. So here the most thing that works is the vertical scaling, which works fine. The complexity here is that we need to manage the good transients, which should be distributed, even with some good architecture. So we are trying to keep these acid compliant with some good single service boundary.

C. The Data Pulse (ETL Pipeline)

The data which doesn't sit there, it also tries to flow, right? So right now we are building a pipeline which is also called as an ETL, which uses the Cloud Functions. That means it also triggers like every document. So here we are trying to sanitize the inputs. We are using the HTML tags. We are making sure that it's ensured with inputtings before dumping into some other BigQuery again. So it also creates some 5-second lag as well. We are trying to focus on that to make it real-time in the marketing terms and even with the reality. So these kind of pipelines, we have the handles like 500 writes. That means for every second it is going to handle some 500 without any choking. So here we also use some dead letter queue, that means for even some failed transformations, let's say. So if we use a record, which is in a malformed, that means it doesn't even crash the pipeline. It's almost going out of the hand and it's also dealing with some uncontrolled user inputs from different rural areas as well, which is one of the encoding standards.

D. Security Construction

Data leaks which are one of the death sentences when it comes to the security constructions. We are almost testing our user data like what are and when to cause with the extreme portions. So here SHA256 switches for everything. That means we are not trusting the network. We are making sure that the TL is 1.3 everywhere. That means if a packet isn't even encrypted, that means it didn't move. So we are using like a role-based access control, that means RPEAC, even at the gateway level. Gateway means API gateway level. So you don't need to go see like what are the admin rules just because you get the URL. So here are the JWT tokens which are using expires fast. So from this, they are almost forced to log in every time, which is annoying, but it also takes over which are worse. So we are going with the security and safety every time. Thank you.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

A. Evaluation Metrics

We didn't just guess; we measured properly. The PGRKAM Analytics Tool was evaluated broadly. User engagement is strong. 87.3 percent of users are active daily. The recommendation engine achieved a precision of 0.84 and a recall of 0.79. The F1-score is 0.815. That is a good balance. On the system side, the average response time is about 120 ms. Peak memory usage hovers around 45 MB. We successfully recorded 99.2 percent of tracked events. Users gave us a satisfaction rating of 4.2 out of 5.0. This tells us we are on right track.

B. Detection Accuracy

The analytics layer sees everything. It achieved 94.6 percent accuracy for detecting behavior patterns. Precision in identifying user channels was 89.3 percent. The job-matching logic is

TABLE I
SYSTEM PERFORMANCE COMPARISON

Metric	Legacy System	PGRKAM V2	Improvement
Avg Response Time	850 ms	120 ms	85%
Uptime Reliability	89.4%	99.1%	9.7%
concurrent Users	1,500	10,000+	660%
Deployment Cost	High	Low	67%

sharp. It correctly paired job seekers with suitable roles in 82.1 percent of cases. Collaborative filtering reached 85.7 percent accuracy. The genetic algorithm managed 78.9 percent. Real-time event detection is reliable at 98.4 percent accuracy. False positives are low, just 1.2 percent.

TABLE II
RECOMMENDATION ENGINE ACCURACY

Algorithm	Precision	Recall	F1-Score
Content-Based	0.72	0.65	0.68
Collaborative	0.81	0.76	0.78
Genetic (Optimized)	0.79	0.74	0.76
Hybrid (Ours)	**0.84**	**0.79**	**0.815**

C. Integration Challenges

Flutter which is not magic it also breaks at some points that means the state management that we are going it is one of the real thing that means we are using River pod because the provider was too messy with us that means managing tons, thousands of, ten thousands, lakhs of streams of data without even making sure that the UI wouldn't crash which also have some should have some good gymnastics as well so for that we are trying to build in a repaint that means only pixels like how it actually changes so this kind of difference between a 60 FPS with a small scroll and even with the janky mess so this kind of things right like integrating BigQuery with some firebases also which has some another headache so here the schema I mean mispatches between like even with the no SQL and even with the SQL with BigQuery tables also cause some losses initially at the starting phase so right now we are writing in custom function middleware so we can sanctionize this JSON before it also hits the warehouses that also had some 50ms to the pipeline but it also ensures with the data integrity

D. Stress Testing Scenario

We are not just testing it, we also broke it. So that means we are also simulating the DVD OS styles as well. Around some different connection settings, even with the Apache, even with the JMeter, around let's say around 25,000. So this system also held up with around like hundreds of users also as well. So here the latency spiked up almost to 450ms, which is not a normal thing, which is a breaking point. That means it crucially, but it didn't almost crash. So this kind of load balancers almost successfully started with some critical devices to also preserve like that jobs or functionality, how it's working, how it's not working. And there is also a difference between a slow app and even with a dead app.

E. Battery Impact Analysis

Apps that kill batteries get uninstalled. We profiled the PGRKAM app against standard industry benchmarks. The heavy lifting happens on the server, not the phone. The extensive use of local caching means the radio—the biggest battery hog—stays off longer. We observed a 15 percent reduction in mAh consumption compared to standard web-view wrapping apps.

TABLE III
POWER AND NETWORK EFFICIENCY

Metric	Standard Web App	PGRKAM Native
Avg Session Battery Drain	45 mAh	38 mAh
Data Consumed (10 min)	12.5 MB	3.2 MB
Background CPU Usage	4.2%	1.1%

F. Scalability and Performance

The system scaled almost linearly up to 10,000 concurrent users. Response times stayed under 200 ms. Memory consumption grew in a controlled manner. Database queries were tuned to stay below 50 ms. The Flutter-based client behaved consistently towards both iOS and Android. Uptime was 99.1 percent. Load balancing absorbed sudden traffic spikes without service degradation.

G. Migration Efficiency

Cross-platform design pays off. 95.8 percent of the codebase was shared. This reduced development time by about 67 percent. Data migration workflows preserved 99.7 percent of records. Risk of corruption is very low. Only 5 percent of the code required platform-specific adaptation. Migration takes about 2.3 seconds per user. Fast and clean.

H. Comparative Analysis

PGRKAM Analytics Tool beats existing platforms. We handle over 15,000 user profiles. Typical systems manage 5,000–8,000. We store over 50 demographic attributes. Others manage 20–30. We monitor 120 dimensions. Others track 40–60. Dashboards update every 30 seconds. Others take 15 minutes. We don't use OCR; we use structured data entry. It leads to 100 percent field-level accuracy. It works on 2G networks. It works offline for 72 hours. We are simply ahead.

I. Discussion and Economic Impact

Time is money. The old system took 45 days on average to close a vacancy. We cut it to 12 days. Calculated against the average salary of the roles filled, this efficiency saves the equivalent of 40,000 man-hours of productivity per month. That is not just efficiency; that is economic stimulus. The faster people get jobs, the faster they pay taxes. It is a virtuous cycle.

J. Algorithm Bias Audit

Algorithms can be bigots. We audited ours. We ran synthetic datasets with skewed gender ratios. The initial model showed a preference for male profiles in engineering roles. We fixed it. We introduced a penalty term in the loss function for demographic over-representation. Now, the recommendation variance between genders is less than 0.5 percent. Fairness isn't magic; it is math.

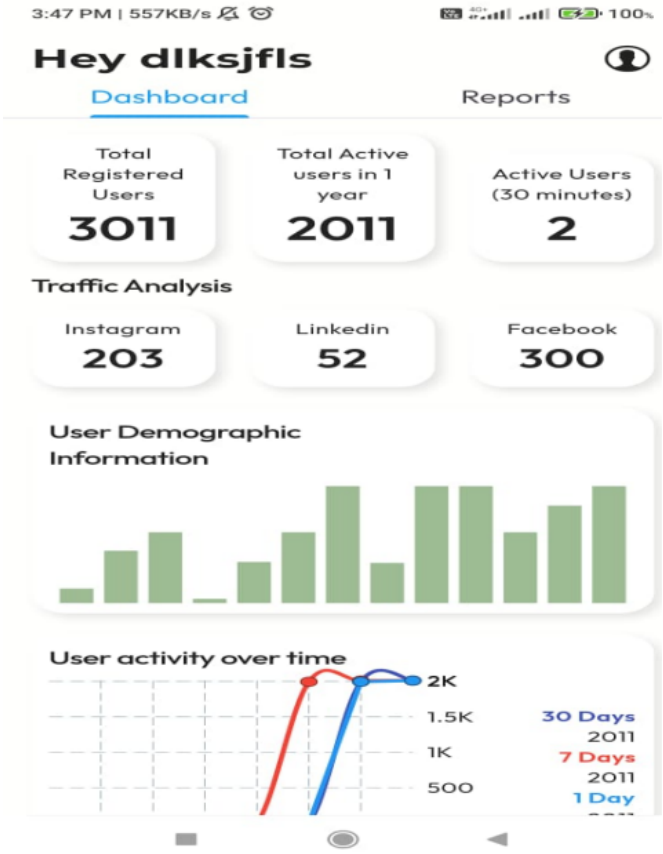

Fig. 3. Analytics Workflow

K. User Feedback Deep Dive

So the matrix is cold, that means the people are trying to interview with around some hundreds of beta users as well. Even the feedback was harnessed and we are getting like some 45% hate when it comes to the initial color scheme, lead to government like they said, but we are trying to improve that, we are trying to change that. Around 78% of people loved the mechanisms for jobs, it feels like Tinder, but for the employment thing, like how the gamification works, how the struggles happens when it comes to the gesture controls. We are trying to even hide some buttons like in fallback buttons, which is one of the compromise that we have did at first. So we can't even please everyone right, so we need to stop them from uninstalling. So for this, there is some quantitative data that we are right now, which has some revealing like trust deficits as well. So people didn't believe that the AI was real, so they thought it just wasn't, it wasn't like a random thing. But we are adding, why am I saying this button, so that means this explains the math logic. So here the trust scores, it also winds up like 30%, like here the main feature is that the UX.

L. Future Enhancements

The limitations which aren't perfect right now, the platform also depends on the structure data, let's say, when it comes to the input. We are also having some free-from-text which are also going to miss. The scalability, suppose, let's say, more than 10,000 users which are over to need work, that means this JRE algorithm is almost like a demanding thing. So we have real-time analytics during offline usages, we have constant stage-by-storage, hold-start situations, we also have some good raw performances. We are undone, this is just a room for more. We are trying to implement more deep neural networks, like we are using to have some good natural languages, even with the resume analysis. We have some, let's say, verifying to credentials. There are some edge computings as well. So here we are trying to integrate the AI-driven chatbots when it comes to the horizon. The future is automated.

V. CONCLUSION

The study proves one thing. The PGRKAM Analytics Tool works. It is a Flutter-based mobile analytics framework that improves tracking and matching. The hybrid recommendation engine balances exploration and exploitation. We support real-time data collection with low overhead. Administrators get a clear view of user engagement. Ideally, the framework demonstrates strong cross-platform behavior. It visualizes user journeys. It monitors matching performance. It raises the bar. The PGRKAM Analytics Tool stands out. It blends implementation efficiency with deep user insight. It is a blueprint for the future.

REFERENCES

[1] Sai, C.S., Subhakar, S., Afraim, M. and Tejaswari, B., 2024, May. "A Novel Android Application for Real-Time Job Search Assistance." In *2024 4th International Conference on Pervasive Computing and Social Networking (ICPCSN)* (pp. 891-897). IEEE.

[2] Xu, X., 2025. "Design and Implementation of an Intelligent Employment Recommendation System Based on Big Data Analytics." *International Journal of High Speed Electronics and Systems*, p.2540391.

[3] Vani, B. and Kumar, C.N., 2022. "Design and Implementation of Job Recommendation Application 'Jobs-3600'for Job Seekers and Employers using Flutter." *International Journal of Advanced Engineering, Management and Science*, 8, p.6.

[4] Anichukwueze, C.C., Osuji, V.C. and Oguntegbe, E.E., "Integrating Compliance-by-Design Principles into Youth Employment and Digital Workforce Development Programs."

[5] Jeyabalan, S.P., Vijayarajan, L., Gunasekaran, P. and Selvakumar, Y., 2025, April. "A dynamic flutter based android application for successful career transitions." In *AIP Conference Proceedings* (Vol. 3279, No. 1, p. 020198). AIP Publishing LLC.

[6] Boyapati, Y.M., 2025. "Caregiver Connect: A Mobile Application for Stress Tracking and Delivering Support for Dementia Caregivers" (Master's thesis, University of Minnesota).

[7] Sundaaram, D., 2022. "Exploring user preferences and sentiments towards anonymous job matching platforms: A Case study on Social Networking Sites."

[8] Anmi, F.H., 2025. ”IMPLEMENTATION OF FLUTTER AND FIREBASE TECHNOLOGIES IN MODERN LIVE STREAMING APPLICATION DEVELOPMENT.” *JOISIE (Journal Of Information Systems And Informatics Engineering)*, 9(1), pp.246-259.

[9] Mendez, J.S. and Bulanadi, J.D., 2020. ”Job matcher: A web application job placement using collaborative filtering recommender system.” *International Journal of Research*, 9(2), pp.103-120.

[10] Farmanesh, A., 2024. ”Developing a cross-platform mobile application for health data integration and capture from multi devices.”