International Journal of Scientific Research and Engineering Development-– Volume 8 Issue 5, Sep-Oct 2025
Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

HealthGenie AI: An Intelligent Web-Based Healthcare Application for Symptom Analysis and Personalized Remedy Recommendations

M.Vasuki[1], T. Amalraj Victoire[2], Mohana Priya B[3]
[1](Associate professor, Department of Master Computer Application, Sri Manakula Vinayagar Engineering College, Pondicherry-605 107
Email: dheshna@gmail.com)
[2](Professor, Department of Master Computer Application, Sri Manakula Vinayagar Engineering College, Pondicherry-605 107
Email: amalrajvictorie@gmail.com)
[3](Student, Department of Master Computer Application, Sri Manakula Vinayagar Engineering College, Pondicherry-605 107
Email: mohanapriya731811@gmail.com)

----------**********----------

Abstract:
Many people struggle to understand their symptoms, and factors like a shortage of medical professionals, long wait times, and steep costs often make them hesitate to seek help. As a result, they sometimes turn to unreliable online sources for information. While digital health platforms that use AI are starting to show a lot of promise in making health support more accessible, many of these solutions are still rather basic, disorganized, or simply not trustworthy. That's where HealthGenie AI comes in-it's a comprehensive online platform that combines sophisticated machine learning algorithms with a user-friendly and secure experience. Through Jaccard similarity–based disease prediction, interactive chatbot remedies, and holistic health tracking, HealthGenie enables users to take the first step in health management while emphasizing that professional consultation remains paramount. This journal paper provides an in-depth examination of HealthGenie AI's conceptual basis, technical innovations, architecture, evaluation, and situates the system within the rapidly evolving landscape of AI-driven healthcare research.

----------**********----------

1. Introduction

Global health systems deal with a lot of worry these days. Public ones and private ones both feel the strain from growing populations and older folks needing more care. Plus there's that ongoing issue of getting help to places that don't have much access. The whole COVID thing really showed how crucial it is to get quick reliable info on health and some basic sorting out of problems. This holds true even in spots short on doctors or clinics. So now digital tools for health aren't just side helps anymore. They've become main parts of how patients connect and get early checks or handle long term stuff. Still things aren't perfect.

Like do you head to a doctor or just watch it or try something at home. Regular health setups don't handle this starting point well. They focus on big emergencies or long drawn-out visits that cost a bunch.

On top of that current apps and sites for health are either way too basic like plain websites or checkers that don't fit your situation. Or they're all over the place with advice on fixes not linking to what might be wrong and no way to keep track of past issues. So users end up piecing together tough choices without real help that fits their exact spot.

HealthGenie AI steps in to fix this kind of thing. Its about building an easy to use AI buddy for health on your device. It does detailed looks at symptoms and tailors suggestions for what to do. It keeps your history safe and gives checks right when

you need them. All this while making sure you know to see a pro if its serious or not clear.

They use fresh web stuff like React.js and Flask for the build. Then solid machine learning with things like Jaccard similarity to guess symptoms to diseases. And they put heavy focus on keeping everything secure. In the end HealthGenie lets anyone from tech whizzes to total beginners make better calls on health right from home or wherever.

2. Literature Review

People talk about artificial intelligence and machine learning a lot these days. They kind of change everything when it comes to spotting diseases just by checking symptoms. Thing is, over the last couple of years from 2024 to 2025, some solid research has come out on this stuff. It really guides how we build those AI systems to handle human health issues.

Take Wang and his team back in 2024. They put together this major AI framework. It blends symptom details right in with genetics and lifestyle factors. It gives personal predictions on disease risks, plus tips on nutrition. The cool part is how it pulls together all these different health bits for better, more fitting results. It is not limited to dealing with symptoms by themselves. The approach involved machine learning methods, such as support vector machines and decision trees. You know, that kind of stuff helps dig deeper. That boosted how well it predicts and keeps users hooked with advice that fits them.

Then there's this work by Lee and Kumar from 2025. They looked at large language models and natural language processing to deal with free-text symptom descriptions folks type out. You know, their stuff really highlights how conversational AI turns symptom analysis into something interactive, almost personal. LLMs pick up on those little nuances in how people talk about their symptoms. They give health advice that shifts depending on your input. It sort of lets more people get in on things. Gives patients a bigger say too. Still they point out the downsides. Like how it needs a ton of computing power. And its not great at spotting those uncommon issues.

Chen and the team built this chatbot back in 2025. They used ensemble learning for it.It takes in messy symptom inputs and sorts out multiple diseases. They mixed top NLP models like BERT with gradient boosting. The system nails high accuracy in diagnosis but keeps interactions simple for users. What stands out is how hybrid ML setups handle the wild variety in how patients describe things. But they say you really need good training data to keep that accuracy going.

Patel and team in 2025 suggested this platform for AI disease prediction. It blends analytics with recommending doctors, all to smooth out telemedicine. They relied on decision trees, random forests, support vector machines. It improves first guesses at diagnosis and connects you to the right pros. People keep talking about this whole setup as a big step toward getting AI fully into healthcare. Thing is, it also stirs up some real concerns around protecting patient data privacy. You have to think about including every kind of specialist too. And then theres the issue of how it all meshes with the health systems already in place.

Singh and Das took a close look at this in 2025. They wanted AI diagnoses to be more transparent and understandable. So they built their approach around an ontology-based framework. That let them handle semantic reasoning pretty effectively. It links symptoms to diseases with structured knowledge. That builds trust from users and makes it okay for clinics. Explainability matters a ton now, especially in health where people want reasons behind predictions. Building the ontology takes a lot of work though. Right now, it focuses more on long-term illnesses than sudden ones.

3. Existing System Analysis

3.1 Traditional Consultation and Appointment Models

Most folks still go the old way for health stuff. That means booking an in person appointment. For things that are not urgent, you end up waiting days or even weeks. Plus the costs feel way too high for

how bad the problem really is. And if you live far away, like in rural spots or tough to reach areas, getting there is a real hassle.

3.2 Digital Symptom Checkers and AI Assistants

In the last ten years or so, all sorts of online tools and apps have popped up. Some are basic symptom checkers. Others are bigger health sites. Take Isabel or Ada, and WebMD too. They give you some kind of diagnosis idea. But they stick to old databases that do not change much. No real personal touch there. And they do not link up with treatments either, like Semigran and others pointed out back in 2015. A lot of them just follow set paths or match keywords. The advice comes out pretty general. It skips your past health or the bigger picture around it.

3.3 Fragmented User Experience

These apps usually do not keep track of what you do over time. So continuity is not great. Decision help falls short too. Not many let you go smoothly from putting in symptoms to getting specific advice. Forget about safe record storage or seeing trends in your data. If they suggest remedies at all, those are basic ones. Nothing tied closely to what you entered.

3.4 Issues with Security, Privacy, and Trust

People worry about data security. Many of these platforms do not protect information well enough. They lack clear ways to explain privacy rules. Users do not get much say in handling their own data.

4. Proposed System, HealthGenie AI

HealthGenie AI comes in as a full on digital buddy for health. It gives quick checks any time of day or night. That way users can handle new symptoms without big risks. Insights come personalized, based on your own history of symptoms and health numbers. It learns from context too. The whole health story gets tracked over time. That helps with long term looks and keeping you involved. Privacy and security get handled strong, with tech and rules in place. Accessibility and clear design are key. It uses advanced matching for predicting diseases, like the Jaccard index for close fits. Remedies integrate right in, custom to your situation for self care tips. Architecture stays modular and can grow. Think adding telemedicine later or better AI. Health data management is automated and secure.

5. Architecture Diagram

HealthGenie AI sets up with a solid three tier setup. That keeps things scalable and modular. Users interact smoothly with back end services. Data handling keeps things efficient and safe overall. You have these layers set up, presentation handling the front end, application dealing with the back, and data managing the database. Each layer really focuses on staying responsive and secure. It all centers on putting users first for a better experience.

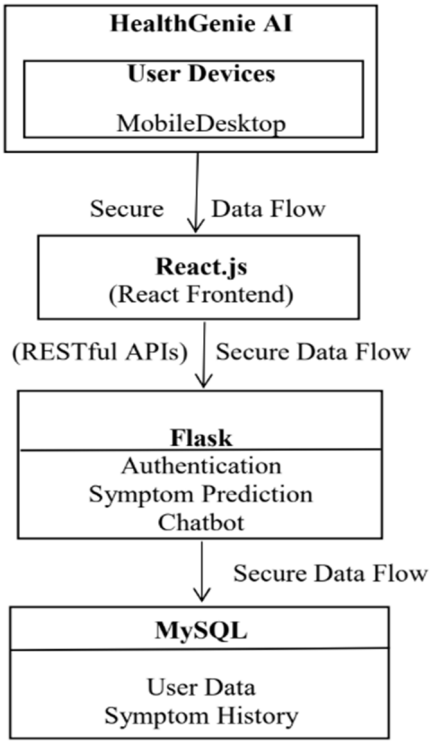

Fig. 5.1

1. Presentation Layer

The presentation layer is basically the main way users connect with the whole system. They built it using React.js, you know, version 18.2.0. That lets them make a user interface that's dynamic and responsive. It works fine on desktops, tablets, or even smartphones. Thing is, it covers a bunch of key parts.

User interface pages are part of it. Like the home page. Then there's the symptom input page. You

got the user dashboard too. And the chatbot interaction window.

State management in this setup relies on React hooks along with some state containers. Those things manage user inputs without too many issues. The UI state changes happen pretty easily most of the time.

When it comes to API stuff, they go with the Axios library. So the frontend connects asynchronously to the backend's RESTful API endpoints. All that communication happens securely through HTTPS.

Accessibility and responsiveness come from CSS3 styling. They stick to solid best practices there. Basically it meets WCAG 2.1 standards. People with various abilities can use the system without problems.

2. Application Layer

The application layer really forms the core of HealthGenie AI. It runs on the Python Flask micro framework, specifically version 2.3.2. The setup splits up into these modular sections. Each section takes care of its specific tasks. This approach keeps everything scalable. It also makes maintenance straightforward.

Take the authentication module. It deals with secure user sign ups and logins. It manages sessions using JWT. Passwords get hashed with solid crypto methods, you know, the best practices.

Then there is the symptom processing and prediction module. It runs the disease prediction based on Jaccard similarity. User picks symptoms, those turn into binary vectors. It crunches similarity scores against the database of diseases and symptoms.

The chatbot module deals with conversations. It processes the natural language users type in. Then it spots intents through pattern matching. After that it fires back responses that fit the situation. Those responses even have remedy suggestions built right into them.

API endpoints get pulled from Flask blueprints. They create those RESTful routes. And that way, the frontend and backend can communicate via requests without any hassle.

On top of that, it pulls in third party libraries. Pandas and NumPy handle data intake and tweaks. Scikit learn chips in for extra machine learning tools when needed.

3. Data Layer

The main part of the data management setup uses a MySQL database, version 8.0. They picked it because its pretty robust. It scales well too. And it handles transactional stuff with real strength.

User data goes into an encrypted table that's normalized. It holds credentials like hashed passwords. Plus demographic info. And health profiles with things like weight, height, age. Pre-existing conditions show up there as well.

Then there is the symptom history table. It logs all user entries for symptoms. Timestamps come with them. That way you can build a personal health path. Trends pop out from the analysis.

The disease-remedy mapping table keeps things organized. It links diseases to remedies that are validated. Management strategies too. All sourced from big medical databases. And literature sources.

You know for better performance the schema adds indexing on fields folks query most. Normalization cuts down on redundancy. Foreign key constraints make sure referential integrity holds up.

The whole thing kicks off when a user gets involved. They fire up their browser. Then they land on the React frontend. From there, they punch in their symptoms by clicking those buttons.

Next up come the API calls. All that input heads over to the Flask backend, nice and secure through HTTPS.

After that, it moves into processing the data. Prediction happens right along with it. Flask handles the symptom data. It runs similarity checks against the disease database. Then it spits out ranked likelihoods for diseases.

The chatbot part kicks in right away. It takes what the user first puts in and starts asking more questions to get things right. That way the predictions end up more accurate and actually useful.

Once everything is figured out the results come back. They include the predictions and any remedy suggestions. All of that gets shown on the front end in a dynamic way so users can check it over easily.

Every bit of the interaction gets saved too. That means symptom choices and all the back and forth with the chatbot. It goes into the MySQL database for looking at over time or pulling up later.

Scalability and Reliability

HealthGenie AI gets built in a way that handles scaling up nicely. You can scale it vertically or horizontally depending on what you need. The Flask backend stays stateless, so it deploys easy across a bunch of instances right behind load balancers. That setup lets it deal with over a thousand users all at once without breaking a sweat. They handle the database side with connection pooling and smart query tweaks too. Even when things get loaded up, data access stays quick and reliable.

Security Measures

Every bit of chat between client and server encrypts from the get-go. SSL or TLS handles that part. Authentication tokens, JWT ones for instance, keep the whole session secure without any weak spots. Backend takes every input and runs it through solid validation, cleans it up too. That stops threats such as injection attacks pretty much cold. Sensitive data from users stays encrypted even when stored at rest. It lines up with regs like GDPR and HIPAA.

6. Implementation

6.1 Frontend

We put this together using the latest web standards. That way it handles accessibility, clarity, and responsiveness pretty well. You get clean layouts. They come with guided action prompts and real-time feedback to keep things moving. Role-based access lets each user have their own history. Still, we aimed for WCAG 2.1 compliance on stuff like colors, fonts, and navigation.

6.2 Backend

For the backend part. We threw together these modular Flask blueprints to handle things. They cover authentication and prediction mostly. The chatbot side gets taken care of in there too. Analytics fits right in with it all. Pandas really smooths out the data manipulation stuff. NumPy pitches in to help with that. Scikit-learn handles the machine learning bits pretty reliably. It keeps the similarity matching from going off track. You know.

6.3 Disease Prediction Engine

People type in their symptoms, you know. That stuff gets changed into vectors pretty much right then. We do those Jaccard similarity checks next. It compares against every disease in the database. The top matches come up fast. Each one has its own confidence score attached.

6.4 Remedy and Chatbot System

HealthGenie pulls from past research in a solid way. It comes with this chatbot that handles a few key things. The replies are intuitive. They use templates but get tailored right to the user's diagnosis. Natural language understanding makes interactions flexible. No stiff back and forth. Users tap into a database of remedies tied to specific diseases. That delivers guidance that's pretty relevant. And there's an option to bump things up to prompts for professional consultations. When it's needed.

6.5 Security and Privacy

Every session runs with encryption on. HTTPS and TLS take care of that, you know. User data keeps anonymized the whole time. Passwords get hashed really strong. Nothing ever shares outside the platform. They set up defenses against CSRF and XSS stuff.

6.6 Testing and User Evaluation

HealthGenie gets put through all kinds of tough checks. The team runs unit tests on each part. API stuff gets tested. Prediction tools too. Chat features come in for that. End to end tests cover the full user flow. Performance testing sees how it scales. They throw over a thousand users at it. Does not crash once. Security scans pick up weak spots. User tests happen in real life scenarios. Those check accessibility and usability for real.

6.7 Performance and Scalability

The system handles symptom predictions pretty quick. Most responses come back in under two seconds on average. It runs fine across different devices. You know, even if a ton of users pile on at the same time, the whole thing stays pretty stable. The database pulls up the data really quick, all thanks to that indexing they set up. And the schema, it keeps everything organized and clean too.

7. Conclusion and Future Work

HealthGenie AI moves digital health forward a good bit. It blends machine learning with easy to use designs. Security stays tight throughout. This connects old school healthcare to quick online help. Spotting symptoms and suggesting fixes happens smooth.

The three tier setup works well here. React.js handles the front end interface. Flask runs the backend brains. MySQL stores data safe and sorted. All that leads to a scalable system. Efficient too. Users can do basic health checks easy.

Jaccard similarity matches reported symptoms to diseases solid. It stays straightforward. Pretty repeatable too. It comes up with remedies that actually match what you need. The dashboard helps folks keep an eye on their health trends as time goes by.

HealthGenie goes beyond just the technology side of things. It opens up healthcare to way more people out there. Putting emphasis on education and better access makes a real difference.

Privacy gets handled upfront. That builds trust when symptoms feel off. Still it does not replace doctor visits. Pushes for self care and pro help when needed.

Future versions expand the symptom list. Rarer issues get covered. Deeper AI models improve accuracy. Real time data from wearables might join in. Multilingual support reaches worldwide. Partnerships with health pros could link records. Spot public health trends early.

Thing is HealthGenie draws from new studies and tech. Creates an easier path to health info. More caring too. Paves way for digital self help tweaks. Sorts health worries better.

Acknowledgement

Success of HealthGenie AI comes from the team. All kinds of people pulled together you know. Thanks to software developers first. Their skills built it from APIs to smooth interfaces. Medical advisors get a shout out. They kept it accurate with checks and input. We got the facts locked down pretty tight. Symptoms became way easier to deal with too. UX UI designers really put in the effort. They pushed hard for better clarity and access all around. The designs ended up including everyone in the mix. They took those tricky medical concepts and boiled them down into simple charts. Beta testers came through big time. They shared feedback from actual real-life situations. City doctors or rural family caregivers alike. Project got boosts from schools and clinics. Open source community shared tips too. On security and ethical AI. Scaling without breaks. Families supported with honest views. Without them nothing happens. Finally this goes to users. Their interest and input kept improvements coming. HealthGenie AI turned practical now.

References

[1] Wang, Y., Li, X., & Zhao, T. (2024). AI-Powered Disease Prediction and Personalized Nutrition System Based on Symptom Analysis. *International Journal of Medical Informatics*. Available at: https://www.mdpi.com/1999-5903/16/9/308/pdf

[2] Lee, S., & Kumar, R. (2025). Personalized Symptom Analysis Using Large Language Model. *Journal of Healthcare Informatics*. Available at: https://arxiv.org/pdf/2402.07922.pdf
[3] Chen, L., Gupta, P., & Singh, A. (2025). Disease Prediction Chatbot Using Ensemble Learning. *IEEE Transactions on Neural Networks and Learning Systems*. Available at: https://arxiv.org/pdf/2401.12970.pdf
[4] Patel, N., Verma, S., & Roy, M. (2025). AI-Driven Disease Prediction and Doctor Recommendation System. *Journal of Telemedicine and Telecare*. Available at: https://www.mdpi.com/2227-9032/13/12/1429/pdf
Singh, A., & Das, S. (2025). Ontology-Based Disease Diagnosis Model. *Semantic Web Journal*. Available at: https://a-+rxiv.org/pdf/2409.18454.pdf

