Volume: 13 Issue: VI Month of publication: June 2025
DOI: https://doi.org/10.22214/ijraset.2025.72239

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue VI June 2025- Available at www.ijraset.com

All Served Service

S. Sriram Lokesh[1], Mrs. D. Nandhini[2]
[2]M.C.A, Assistant professor, Department of Computer Applications, Christ college of engineering and Technology, Puducherry – 605010

Abstract: ***This project is all about making home services... less painful. Imagine this—you've got a leaking pipe, wires acting weird, or your washing machine suddenly gives up. What do you do? Scroll through contacts? Call random numbers? Nah. This web app got you.***
Built using HTML, CSS, JavaScript, Django, and SQLite. It brings together customers and service providers—like plumbers, electricians, cleaners, and repair experts—on one simple platform. Register quickly. Book a service. Done.
The app handles everything. Real-time service updates, secure logins, smooth UI. It's like the modern version of calling the handyman, but without the wait or the guesswork.
It's formal where it should be. Chill when it can be. Designed to be scalable, easy to use, and just... smart.
The goal? Digitalize the old ways. Make home services faster, easier, and a whole lot better. No more stress. Just help—when you need it.
Key Points:
- ***Frontend – Made with HTML5, CSS3, JavaScript [3][4]. No React. No bloat.***
- ***Backend – Django's doing the heavy lifting [1]. Powerful, reliable.***
- ***Database – SQLite, simple and silent [2].***
- ***Authentication – Yup, that's in [1][9].***
- ***Service Booking – Dead simple [1][10].***
- ***Admin Panel – Neat and solid [1][10].***

I. INTRODUCTION

In today's fast-moving digital age, nobody has time to wait. Pipe bursts. The AC stops working. Or maybe your fridge just... died. You don't wanna dig through old phone books or beg your neighbor for contacts. You need help. Fast.
That's why this project exists. It's not just a web app. It's your go-to place when stuff goes wrong at home. Need a plumber? Done. Cleaning crew? Booked in seconds. One platform. Few clicks. No chaos.
Tech-wise, Django's the brain. Reliable. Fast. Gets the backend job done without fuss. Frontend? That's all HTML, CSS, JavaScript—smooth, simple, gets the job done. SQLite keeps the engine light. It tracks users, services, and bookings. Doesn't complain.
This app? It's built for real life. For messy schedules. For "I need this fixed right now" moments.
It doesn't just solve a problem. It makes life easier. Cleaner. Quicker. No fancy fluff. Just what you need—when you need it.

II. ALGORITHM

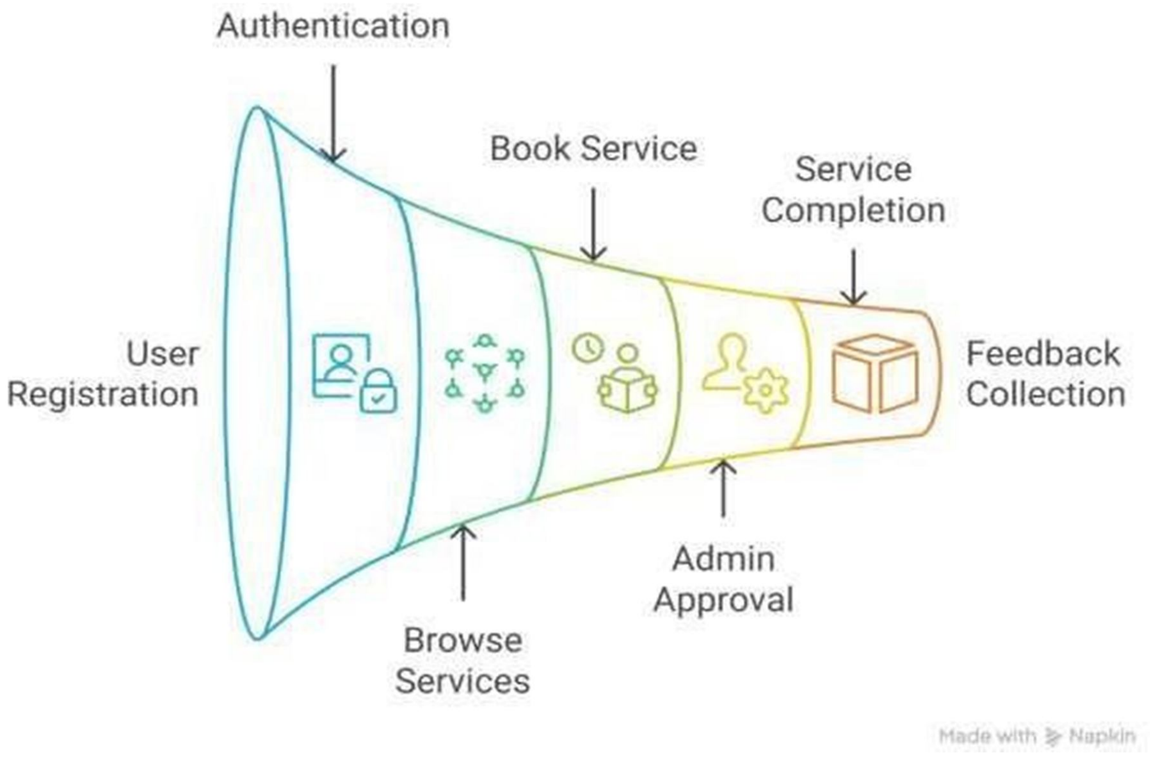
1) User Registration & Login
The user fills out a form with the usual details: name, email, and password.
Django takes over. Handles auth like a boss. Stores everything neatly in SQLite. Boom, you're in.

2) Browse Services
The user lands on the dashboard. Services are available—plumbing, cleaning, repairs, and all that good stuff.
Pulled straight from the database. No reloads. Just smooth, dynamic content.

3) Book Service
Pick a service. Choose a time that works.
The system checks it over. If it's all good? It gets saved. Instantly. You just booked help without making a single call. Nice.

4) Admin Approval (if needed)
Sometimes, the admin jumps in. Gives the green light or assigns a provider. Sometimes not. Depends on how it's set up. Either way, you're covered.

5) Service Completion & Feedback
The provider shows up. Gets the job done.
Afterward, you rate 'em. Maybe drop a quick review. Helps others. Feels good.



III. ADVANTAGES

It's simple. Really simple. You tap, it works. Fast pages. Clean buttons. No spinning wheels or weird tech words. Just—book it. In bed, at lunch, wherever. Same smooth ride every time. Feels natural. Like, why wasn't it always this way?
Behind the curtain? Django's doing magic. Quiet. Reliable. Fast like it's got somewhere to be. Want more stuff later—like maps, payments? Sure, throw it in. SQLite's chill. Holds all your data—users, bookings, everything—without complaining. Your info? Safe. Locked up tight. Login, logout, reset—done. You won't even think about it.
The booking part? Whole thing's wrapped up neat. No guessing who's free. It's just there. Services, times, all listed out. You pick, tap confirm, boom—it's set. Real-time updates? Of course. No awkward no-shows. Admins? They're in the back, running the show. You don't see it. You just feel how smooth it all goes.

IV. FUTURE ENHANCEMENTS

Future's looking cool. First up—mobile app. Built with Flutter or React Native. So yeah, users can book stuff right from their pocket. On the move. In a cab. Or while waiting for coffee. Just tap, book, done. Clean UI. Same feel as the web. But faster. Closer.
Next big thing? Online payments. Razorpay, Stripe, maybe both. No more cash. Just tap to pay. Feels modern. Feels safe. Add in live chat while we're at it. So users can talk to providers. Ask stuff. Get updates. Maybe send a pic like "hey, this is broken—can you fix this?" Super handy.
There's also talk about going smart. Like, using GPS to match services by location. You need someone close? The system finds them. Automatically. Add push notifications too— reminders, updates, "your cleaner's on the way!" type stuff. And later? Machine learning. Recommending stuff based on what you booked before. Feels personal. Kinda like the app *knows* you. Creepy? Maybe a little. But also… kinda awesome.

V. CONCLUSION

So here’s the deal. Finding reliable home services? It's been a pain for too long. This project flips that. One platform. All in one place. No calls. No waiting around. Just click, book, done. Django runs the backend—strong and steady. SQLite keeps the data tight. HTML, CSS, and a sprinkle of JavaScript make it all look smooth. It works. Really well. And if it doesn’t? We make it better. But this ain’t the final version. Not even close. With mobile apps, payments, and smart features on the roadmap, it’s only gonna grow. One step closer to a full-blown service platform people actually wanna use. And trust. That’s the goal.

REFERENCES

[1] Django Software Foundation. Django Documentation (4.x). https://docs.djangoproject.com/
[2] SQLite Consortium. SQLite Documentation. https://www.sqlite.org/docs.html
[3] Mozilla Developer Network (MDN). HTML5, CSS3, and JavaScript Resources. https://developer.mozilla.org/
[4] W3Schools. Web Development Tutorials: HTML, CSS, JavaScript. https://www.w3schools.com/
[5] Razorpay. Razorpay Payment Gateway Docs. https://razorpay.com/docs/
[6] Stripe. Stripe API Reference. https://stripe.com/docs/api
[7] Google Developers. Flutter Documentation. https://flutter.dev/docs
[8] React Native Team. React Native Docs. https://reactnative.dev/docs/getting-started
[9] OWASP Foundation. Authentication Cheat Sheet. https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html
[10] GeeksforGeeks. Build a Booking System with Django. https://www.geeksforgeeks.org/