

## INTRODUCTION

- Junior IT professionals often struggle to gain real-world experience, while existing platforms are built around full-time hiring or reputation-based freelancing.
- FirstByte connects Clients with small, discrete IT tasks to students and junior professionals seeking practical, hands-on experience. prior experience — is the entry point.

## METHODOLOGY

- Iterative, build-test-refine development: each feature implemented, tested in the browser, and confirmed working before the next began.[1]
- Built with Python, Django, MySQL (migrated from SQLite), HTML/CSS, and VS Code.[2]
- Database: SQLite migration to fix MySQL.

## KEY FEATURE: Tiered Task-Matching

- Professionals declare an experience level (Entry / Intermediate / Advanced); Clients set a required level for each task.
- Job listings are filtered so a Professional sees only tasks at or below their declared level.

Registration form showing Role and Level selection.

Role: Professional

Level: Entry

Required for Professionals — your experience level.

Register

Figure 2: Role and experience-level fields at registration form

```
11 def job_list(request):
12     jobs = JobPost.objects.filter(status='OPEN').select_related('category')
13
14     # Level matching: professionals only see jobs at or below their level
15     LEVEL_ORDER = {'ENTRY': 1, 'INTERMEDIATE': 2, 'ADVANCED': 3}
16     profile = getattr(request.user, 'profile', None)
17     if profile is not None and profile.role == 'PROFESSIONAL' and profile.level:
18         my_rank = LEVEL_ORDER.get(profile.level, 1)
19         allowed = [lvl for lvl, rank in LEVEL_ORDER.items() if rank <= my_rank]
20         jobs = jobs.filter(level__in=allowed)
21
```

Figure 3:Level -matching logic in the job listing view (job\_list)

### REFERENCES:

- [1] C. Larman and V. R. Basili, "Iterative and Incremental Development: A Brief History," *Computer*, vol. 36, no. 6, pp. 47–56, Jun. 2003, doi: 10.1109/MC.2003.1204375.
- [2] Van Rossum, G. (2009) — "Python Programming Language", *ACM SIGPLAN Notices*, vol. 44, no. 7, pp. 16–27.
- [3] Kshetri, N., Kumar, D., Hutson, J., Kaur, N., & Osama, O. F. (2024). "algoXSSF: Detection and analysis of cross-site request forgery (XSRF) and cross-site scripting (XSS) attacks via Machine learning algorithms." arXiv:2402.01012.

## CASE DIAGRAM

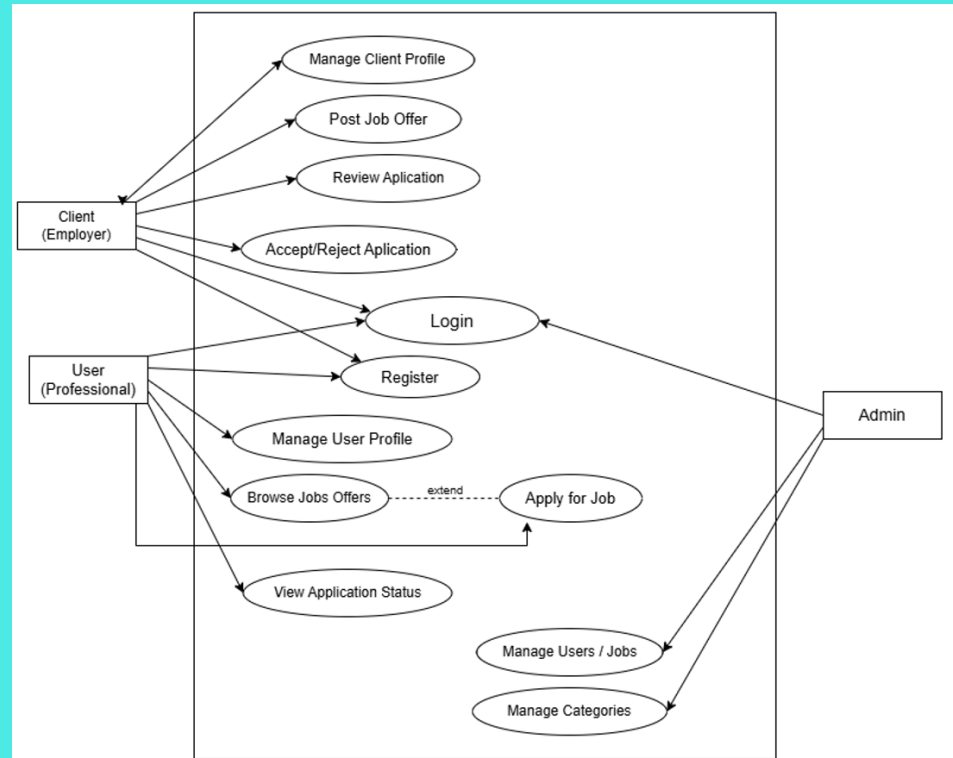


Figure 1: Case Diagram. Users flow interactions

## SECURITY

- SQL Injection: prevented via Django's ORM, no raw SQL used anywhere in the project.
- CSRF: Django's token-based protection applied to every form.
- XSS: Django's template engine auto-escapes all rendered content by default.[3]

Login form showing a successful login attempt.

Login

Please enter a correct username and password. Note that both fields may be case-sensitive.

Username: ' OR '1'='1

Password:

Login

Figure 4: SQL Injection attempt in user login form

## EVALUATOIN

- Stated objectives met: role-based registration, job posting, applications, accept/decline, and level-matching all function as intended.
- Testing combined manual feature testing, end-to-end system testing.
- Limitations: self-declared experience levels (no verification) and reliance on manual no automated testing.