MALNAD COLEGE OF ENGINEERING

Under the auspices of M.T.E.S

(An autonomous institution under Visvesvaraya Technological University, Belgaum)
Hassan – 573201, Karnataka, India

Report on

Course Title: Full Stack Development

“Interview preparation portal”

Submitted by team number :11

Name	USN
Prem Vatage	4MC23IS079
Shivraj S M	4MC23IS099
Tejas Kumar D H	4MC23IS115
Vidyasagar Mahaveer Navalagund	4MC23IS124

Submitted to

Mr. Krishna Swaroop A

(Assistant Professor Dept. of ISE)

Department of Information Science & Engineering

Malnad College of Engineering

Hassan– 573202

2025-26

Date of Submission:

Index

Section/Contents **Page Number**

1. Abstract

The Interview Portal is a comprehensive Django-based web application designed to enhance student preparation for aptitude, technical, and HR interview rounds. The platform addresses the challenges faced by learners who depend on scattered online resources by offering a structured and engaging environment with 60 well-curated MCQs distributed across Aptitude, Technical, and HR categories. It includes two intelligent learning modes—Study Mode for concept reinforcement and Quiz Mode for timed practice—simulating real interview conditions. Features such as a countdown timer for aptitude quizzes, auto-submission, and intuitive scoring mechanisms ensure that the practice experience closely mirrors competitive examinations.

A notable highlight of the system is its integration of performance analytics and gamification. The dashboard uses Chart.js visualizations to display score trends, accuracy levels, topic-wise strengths, time analysis, and long-term learning progress. This data-driven feedback enables learners to identify weaknesses and receive personalized practice recommendations. The gamification layer further enhances user engagement by awarding points, badges, and streak bonuses based on consistency, speed, and accuracy. Badges such as “Algorithm Ace,” “Streak Keeper,” and “Speed Runner” motivate students to practice more regularly and improve their performance. HR practice is strengthened through an audio recording feature that encourages students to assess and refine their communication skills.

The system has been built using modern technologies including Django, Tailwind CSS, Chart.js, and SQLite, ensuring a secure, scalable, and responsive user experience. Dark mode support, CSRF protection, structured model relationships, and modular app design demonstrate adherence to good software engineering principles. Overall, the Interview Portal offers an all-in-one, interactive preparation ecosystem that combines structured learning, analytics, and motivation—making it an effective tool for students preparing for campus placements and professional interviews.

2. Introduction

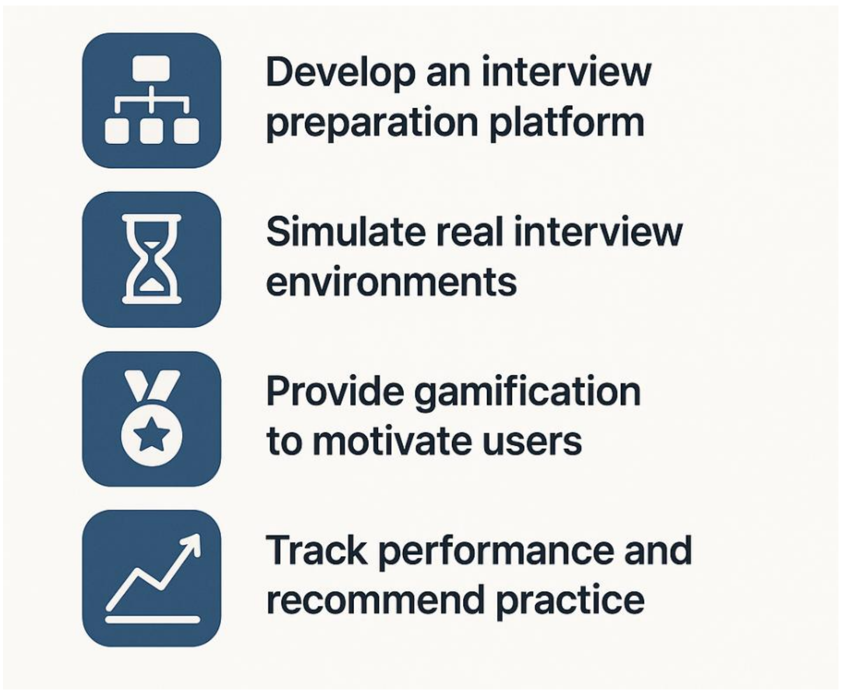
The process of preparing for placements and competitive interviews has increasingly become multidimensional, requiring proficiency across aptitude problem-solving, technical domains, and soft skills such as communication. In modern placement ecosystems, students often struggle due to fragmented study resources, untracked progress, insufficient feedback, and lack of motivation. Traditional preparation methods—textbooks, unstructured online quizzes, and passive videos—do not provide integrated analytics or personalized practice recommendations, leading to ineffective preparation.

The Interview Portal was developed to address these challenges by creating a unified digital platform that supports structured learning and automated performance evaluation. The project emphasizes real-world educational scenarios faced by college students, allowing them to practice in an environment that closely simulates actual interview conditions. The inclusion of audio-based HR practice replicates face-to-face interview rounds, helping users improve clarity, confidence, and articulation. The system also overcomes issues like the absence of records in existing solutions by storing detailed attempt histories, which power the performance dashboard and recommendation engine.

The proposed solution stands out by incorporating gamification to increase learner motivation and engagement. By awarding points, badges, and streak rewards, the platform encourages consistent practice behaviour. The system design further includes robust security mechanisms such as CSRF protection, secure authentication, and controlled media uploads, ensuring safe and reliable operation. Thus, the Interview Portal represents a complete framework for interview preparation, combining pedagogy, analytics, and technology in a cohesive learning application.

3. Objectives

The primary objective of this project is to develop a full-stack interview preparation platform that enhances student readiness for campus placements and professional interviews. The system aims to streamline learning by offering structured practice materials, automated quizzes, performance analytics, and skill-based recommendations. Another objective is to simulate real interview environments through features such as auto-timed quizzes and audio-based HR interactions, thereby improving both technical and soft skills. A key focus of the project is to provide a gamified ecosystem that motivates students through rewards and streak-based encouragement. Additionally, the system aims to maintain secure and error-free operations using robust backend validation, secure authentication, and consistent data storage mechanisms. Finally, the project intends to serve as an extensible foundation that can later incorporate AI-driven feedback, additional question categories, and advanced analytics.


Develop an interview preparation platform
Simulate real interview environments
Provide gamification to motivate users
Track performance and recommend practice

4. System Requirements

4.1 Software Requirements

The Interview Portal is built using a modern and efficient software stack centered around Python 3.10+ and Django 4.2, which together provide a stable, secure, and scalable backend framework. Django's Model–View–Template architecture makes it suitable for handling authentication, quiz logic, database interactions, and gamification workflows. SQLite is used as the development database due to its simplicity and serverless structure, making it ideal for academic projects and local testing. The frontend interface is constructed using Tailwind CSS via CDN, ensuring responsiveness, fast loading, and reduced dependency management. Chart.js is integrated to generate interactive performance visualizations, while JavaScript enhances functionalities such as quiz timers, dark mode toggling, and HR audio recording using the Web Audio API. The project also uses Pillow for image processing and media handling. Optional tools such as Celery and Redis can be incorporated for asynchronous tasks in large deployments. Overall, the software requirements are lightweight and accessible while supporting upgrades to more robust production environments like PostgreSQL, Nginx, Gunicorn, and cloud-hosted storage solutions.

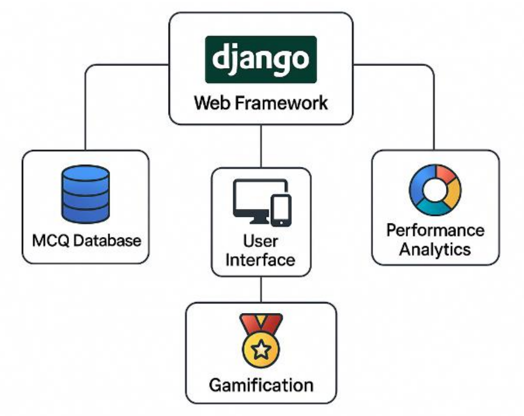
4.2 Hardware Requirements

The hardware requirements for the Interview Portal are minimal, making the system suitable for development on standard student laptops or desktops. A machine with at least a dual-core processor and 4GB RAM is sufficient to run the Django development server smoothly, though using 8GB RAM and an SSD provides noticeably improved performance during migrations, testing, and running multiple services. The application does not demand high-end hardware since most processing—such as quizzes, dashboards, and UI rendering—is lightweight and handled efficiently by the browser and Django backend. For deployment or hosting, a modest cloud server with 2–4 virtual CPUs and 4–8GB RAM is adequate to support moderate traffic, especially if paired with a dedicated PostgreSQL database instance and cloud storage for audio recordings. Reliable network connectivity is recommended during both development and deployment to handle library installations, CDN access, and media uploads. Overall, the hardware requirements are flexible and scalable, ensuring compatibility with both low-spec development machines and more powerful production servers.

5. System Design

The system adopts a layered architectural style, dividing the application into client, server, database, and static resource layers for clarity and maintainability. The client layer focuses on rendering templates, managing UI events, running the quiz timer, recording HR audio, and loading charts dynamically. The Django server layer handles the main business logic—authentication, quiz creation, data processing, scoring, streak tracking, and badge allocation. Models define the core data structure using Django ORM, enabling clean mappings between Python classes and database tables.

The database layer stores structured information such as questions, options, quiz attempts, performance metrics, and badge achievements. Meanwhile, static assets served through CDNs minimize latency and enhance performance. The system's design ensures seamless interaction where the client sends user requests, Django processes and communicates with the database, fetches static resources when needed, and returns responses back to the client. The use of AJAX and asynchronous JavaScript allows real-time updates without full page reloads.

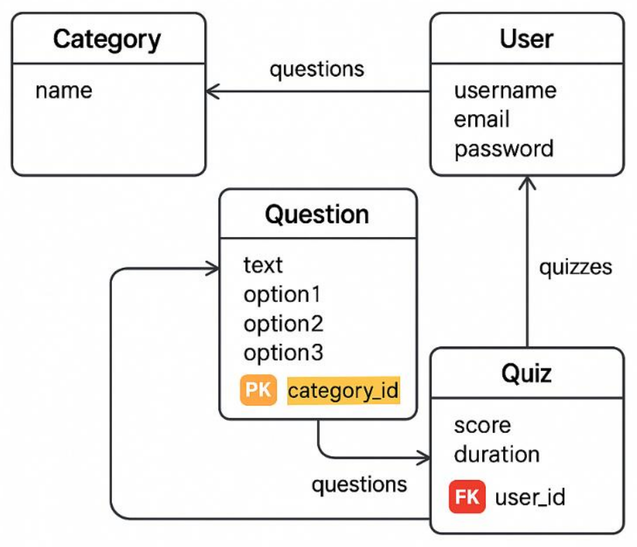


The architecture is further enhanced with security layers that include CSRF protection, input validation, file upload verification, and strict access control policies. Scalability considerations ensure that the system can accommodate thousands of users by upgrading the database, caching frequent queries, and deploying Django behind a load balancer. Overall, the system design blends robustness, modern UI/UX, and modular engineering practices.

6.Database Design

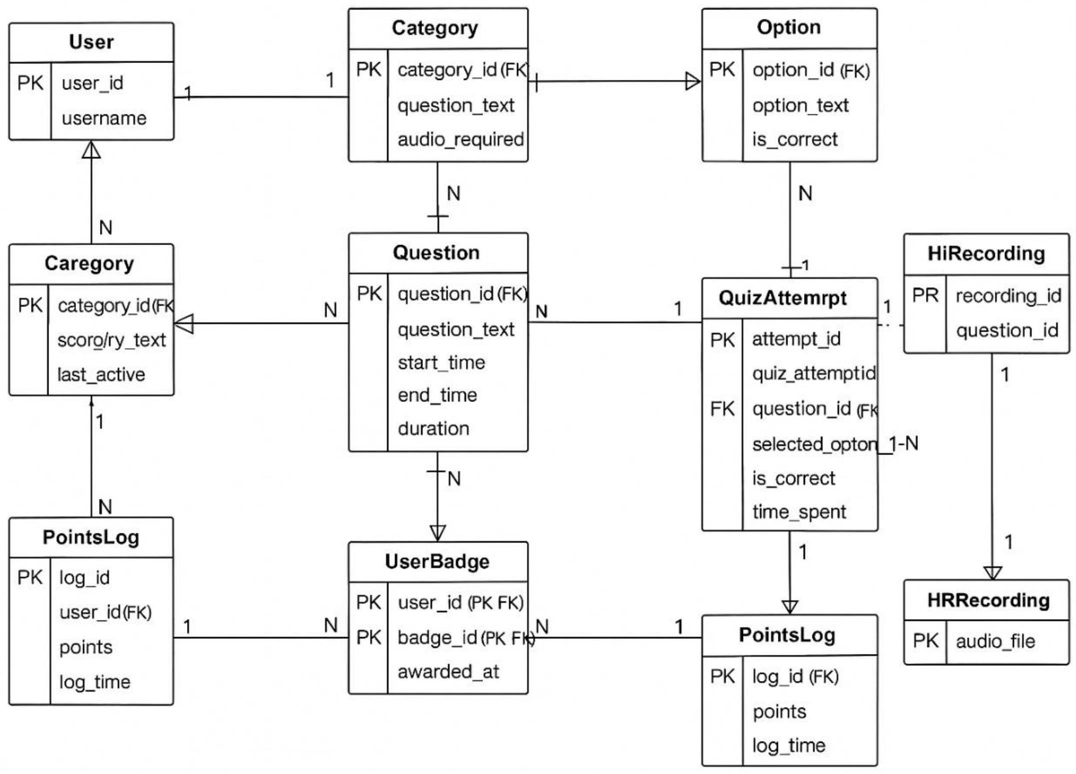
The database design of the Interview Portal is structured using a normalized relational model to ensure consistent, efficient, and reliable data handling across all features of the application. At the foundation of the schema is Django's built-in User table, responsible for securely storing essential user credentials. This is extended by a custom User Profile model, which holds additional attributes such as accumulated points, streak counts, badge achievements, and last-active timestamps. Separating authentication data from profile and gamification data maintains modularity and ensures that user-specific enhancements can be added without modifying core authentication logic. The use of foreign keys ensures referential integrity between user accounts and their associated activity records.

The core interview preparation content is managed through interconnected tables such as Category, Question, and Option. Each question belongs to a specific category—Aptitude, Technical, or HR—and maintains a one-to-many relationship with its options. The Option table stores each possible answer along with a flag marking whether it is correct, enabling precise evaluation during quizzes. To support detailed analytics and performance evaluation, the system uses Quiz Attempt and Question Attempt models, where each quiz attempt stores overall score, duration, and completion details, while each question attempt records the selected answer, correctness, and time spent. This structure allows the application to generate score history graphs, topic-wise proficiency analysis, and time-based performance metrics.



Gamification features are handled through dedicated tables such as Badge, UserBadge, and Points Log. The Badge table defines the achievements available to users, while the UserBadge model records which badges a user has earned and when. The Points Log table captures every points transaction, ensuring transparency and easy integration with leaderboards and dashboards. The database also supports HR practice through optional media-linked models that store audio metadata tied to user attempts, while the actual media files are stored separately to maintain efficiency. With all major tables normalized up to Third Normal Form, the schema avoids redundancy, supports fast queries through indexing, and works seamlessly with Django's ORM to deliver a scalable, maintainable data foundation for the entire Interview Portal system.

6.1 ER Diagram



8. Implementation

The implementation integrates Django views, templates, forms, and models to create a cohesive workflow. The quiz system was implemented using custom views that generate attempts, store progress, and compute scores based on the user's responses. A JSON-based API supplies data for mobile-friendly charts, while the frontend uses Chart.js to render interactive graphs. Tailwind CSS ensures consistent design across pages with minimal custom styling. JavaScript components manage the timer and audio recording interface for HR practice. The application follows Django's best practices such as MVT pattern, secure form handling, reusable templates, and database migrations for incremental schema updates.

8.1 Models overview

The data model is organized around a small set of cohesive, well-documented Django models that map directly to the project's core features. UserProfile extends Django's built-in User model to store gamification-specific attributes (points, streak_count, last_active), profile metadata, and settings such as dark-mode preference. Category, Question, and Option form the MCQ content model: Question records the prompt, metadata and a flag for audio-required HR items while Option holds answer choices and a boolean is_correct marker. Attempt tracking uses two linked models: QuizAttempt (captures start/end times, category, total score and duration) and QuestionAttempt (records the chosen option, correctness, and time_spent per question). Gamification is represented by Badge and UserBadge (award history) plus a PointsLog table that auditable logs every points change. For HR practice, HRRecording stores metadata for audio files while actual media is stored on disk or cloud storage. Models are kept normalized, use meaningful indexes on foreign keys and timestamps, and include helpful model methods to encapsulate scoring and badge-award rules so business logic remains testable and centralized.

8.2 URL routing overview

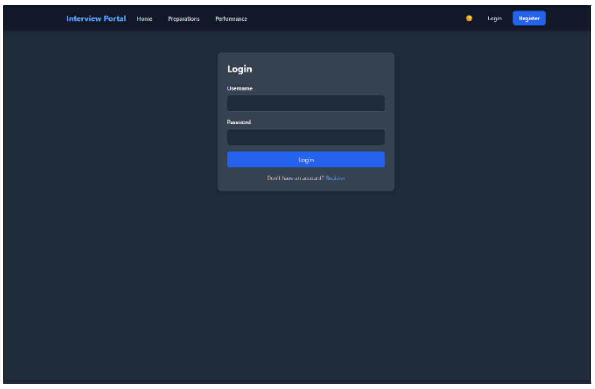
URL design follows RESTful and readable patterns organized by app. Core routes include the dashboard (/), study and question browsing (/questions/study/<category>/), quiz lifecycle endpoints (/quiz/start/<category>/, /quiz/next/, /quiz/submit/), HR practice endpoints (/hr/record/, /hr/upload/), performance APIs (/api/performance/score-history/, /api/performance/topic-proficiency/) and account flows (/accounts/login/, /accounts/logout/,

/accounts/profile/). Routes that perform state changes (quiz submit, audio upload) are POST-only and protected by CSRF tokens; JSON API endpoints return compact payloads consumed by the client JS for charts and live updates. The routing layout groups related views into the preparations, performance, gamification, and accounts apps for clarity, making it straightforward to mount app URLs in the project urls.py and to apply per-app permission decorators or DRF viewsets when needed.

8.3 Important functionality

Aptitude Quiz with Timer and Auto-submit: Starting a quiz creates a QuizAttempt and the frontend runs a countdown. The client autosaves answers periodically and the server validates timestamps on submission; if the timer expires the server finalizes the attempt and scores whatever answers were received. Study Mode vs Quiz Mode: Study Mode serves full questions with explanations and no timers; Quiz Mode randomizes questions, enforces time limits (where applicable), and records QuestionAttempt entries for analytics. HR Practice with Audio Support: Client uses the Web Audio API to record audio, compress to a web-friendly format, and upload to a protected endpoint; the backend validates file size/type and stores a metadata record (with file path) for later review. Performance Dashboard & Charts: Aggregation endpoints compute score history, topic proficiency and time analysis; Chart.js consumes these JSON endpoints to draw line, radar and bar charts on the client. Gamification Engine: Scoring rules are centralized — +10 per correct answer, +20 quiz completion bonus, +5 daily streak — and the badge engine runs after quiz completion to evaluate criteria (e.g., ≥20 technical correct → Algorithm Ace). Points changes are inserted into PointsLog for traceability. Weakness Detection & Recommendations: A lightweight rule engine analyzes recent QuestionAttempt data to mark topics with low accuracy and surface them as "recommended practice" cards on the dashboard. Security & Robustness: All modifying endpoints require authentication and CSRF; file uploads are validated; server enforces request timing checks to prevent fraudulent late submissions; background tasks (audio transcoding or expensive aggregates) are pushed to asynchronous workers when available to keep web responses snappy.

9. Screenshots

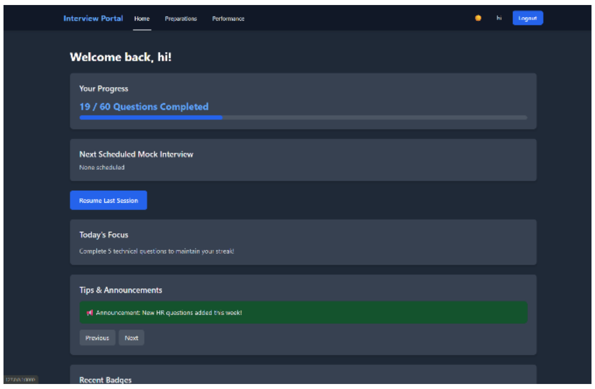


Login

Username

Password

Don't have an account? Register



Welcome back, hi!

Your Progress

19 / 60 Questions Completed

Next Scheduled Mock Interview

None scheduled

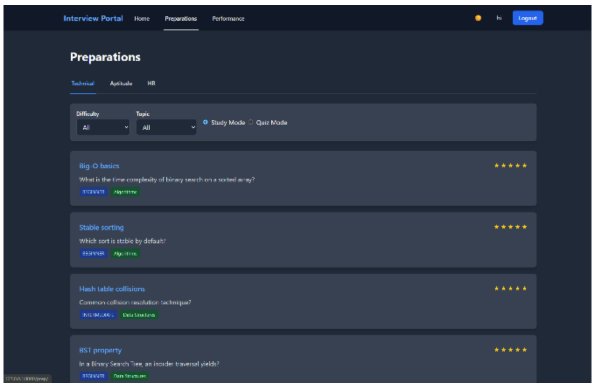
Today's Focus

Complete 5 technical questions to maintain your streak!

Tips & Announcements

📢 Announcement: New HR questions added this week!

Recent Badges



Preparations

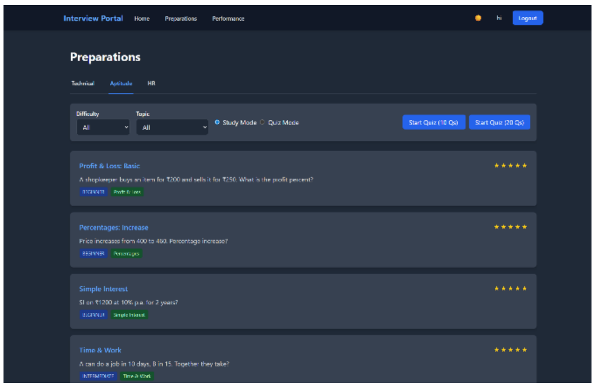
Difficulty: All · Topic: All · ● Study Mode ○ Quiz Mode

Big-O basics ★★★★★
What is the time complexity of binary search on a sorted array?
BEGINNER · Algorithms

Stable sorting ★★★★★
Which sort is stable by default?
BEGINNER · Algorithms

Hash table collisions ★★★★★
Common collision resolution technique?
INTERMEDIATE · Data Structures

BST property ★★★★★
In a Binary Search Tree, an inorder traversal yields?
BEGINNER · Data Structures



Preparations

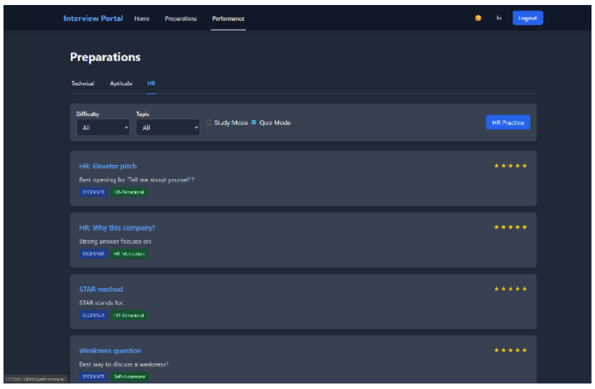
Difficulty: All · Topic: All · ● Study Mode ○ Quiz Mode

Profit & Loss: Basic ★★★★★
A shopkeeper buys an item for ₹200 and sells it for ₹250. What is the profit percent?
BEGINNER · Profit & Loss

Percentages: Increase ★★★★★
Price increases from 400 to 460. Percentage increase?
BEGINNER · Percentages

Simple Interest ★★★★★
SI on ₹1200 at 10% p.a. for 2 years?
BEGINNER · Simple Interest

Time & Work ★★★★★
A can do a job in 10 days, B in 15. Together they take?
INTERMEDIATE · Time & Work



Preparations

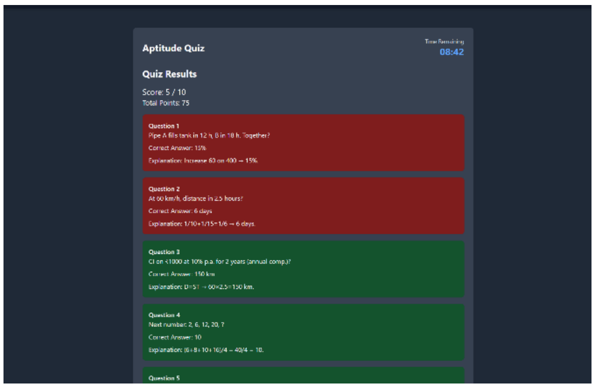
Difficulty: All · Topic: All · ○ Study Mode ● Quiz Mode

HR: Elevator pitch ★★★★★
Best opening for 'Tell me about yourself'?
BEGINNER · HR-Behavioral

HR: Why this company? ★★★★★
Strong answer focuses on:
BEGINNER · HR Motivation

STAR method ★★★★★
STAR stands for:
BEGINNER · HR-Behavioral

Weakness question ★★★★★
Best way to discuss a weakness?
BEGINNER · Self-Awareness



Aptitude Quiz

Time Remaining 08:42

Quiz Results

Score: 5 / 10
Total Points: 75

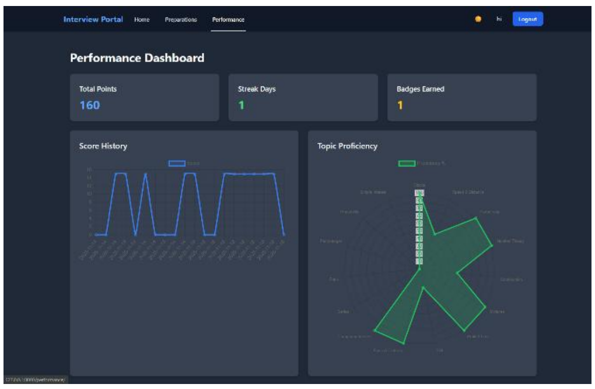
Question 1
Pipe A fills tank in 12 h, B in 18 h. Together?
Correct Answer: 15%
Explanation: Increase 60 on 400 → 15%.

Question 2
At 60 km/h, distance in 2.5 hours?
Correct Answer: 6 days
Explanation: 1/10+1/15=1/6 → 6 days.

Question 3
CI on ₹1000 at 10% p.a. for 2 years (annual comp.)?
Correct Answer: 150 km
Explanation: D=ST → 60×2.5=150 km.

Question 4
Next number: 2, 6, 12, 20, ?
Correct Answer: 10
Explanation: (6+8+10+16)/4 = 40/4 = 10.

Question 5

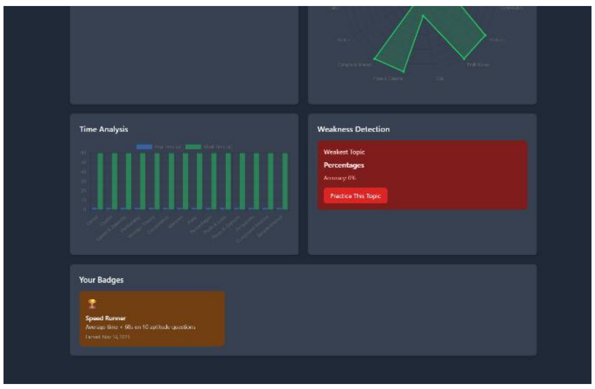


Performance Dashboard

Total Points 160

Streak Days 1

Badges Earned 1



Weakness Detection

Weakest Topic
Percentages
Accuracy: 0%

Your Badges

🏆 Speed Runner
Average time < 60s on 10 aptitude questions
Earned Nov 10, 2025

10. Testing

Testing involved validating functionality across authentication, quiz flow, scoring logic, badge awarding, audio uploads, and analytics retrieval. Form validation tests ensured the system rejects incomplete submissions or malformed input. Security testing included verifying CSRF tokens, enforcing access control, and handling file uploads safely. Edge-case tests included quiz auto-submit when the timer ends, handling rapid navigation, and verifying streak increment logic across consecutive days. Admin panel tests validated CRUD operations for questions and user management. Analytics endpoints were tested using sample data to ensure accurate chart rendering. The testing phase confirmed that the system performs reliably across devices and browsers.

10.1 Form validation

For every user-facing form (signup, profile edit, question creation, quiz submission, audio upload) validate both client- and server-side behaviour. Test cases should include: required-field checks (submit with each required field blank → expect validation errors shown and HTTP 400 on AJAX submits), type and format checks (invalid email formats, overly-long strings, non-numeric input where numbers expected → expect friendly error messages and rejected submissions), and cross-field validation (password + confirm mismatch, start_time/end_time logic for quizzes → expect rejection and clear message). Also test file uploads: too-large files, unsupported MIME types, and corrupted payloads — server should reject with an appropriate status and not store the file. Edge cases include injection attempts (special characters, SQL-like payloads) and concurrent submissions; assertions should verify no backend exception is thrown, errors are user-friendly, and no invalid records are created. Automate these with unit tests for form serializers and integration tests (pytest + Django test client) plus a small set of end-to-end checks with Playwright/Selenium for client-side error UI.

10.2 Login (Authentication & Authorization)

Verify correct handling of all authentication flows: successful login (valid credentials → 200/302 and session created), invalid credentials (wrong password/email → show non-descriptive error, no session), blocked or inactive users (cannot login), and password reset flows (request → receive token behavior simulated, token expiry handling). Test session behavior: logout should destroy session, session cookie flags (Secure, HttpOnly) should be present in production-like settings, and persistent-session "remember me" flows should respect expiration. Test authorization by attempting to access protected endpoints (performance dashboard, quiz submission, admin APIs) as anonymous and as authenticated users — expect redirects or 403 as appropriate. Include brute-force mitigation checks (rate-limiting responses) and multi-tab session consistency. Automate with integration tests that assert session cookies and permissions; include unit tests for custom authentication helpers.

10.3 CRUD operations

Validate Create, Read, Update, Delete flows across core models: Questions/Options/Category (content management), UserProfile edits, QuizAttempt and QuestionAttempt records, and Badge/UserBadge management. For each model verify happy path (create valid record → readable in list/detail view), update (partial and full updates persist correctly), and delete (soft-delete vs hard-delete behavior as designed; if soft-delete is used verify it is excluded from queries but present for audits). Check concurrency: two users editing the same question should not corrupt data; optimistic locking or last-write-wins behaviour should be documented and tested. For data integrity check foreign-key constraints (deleting a Category with Questions should either reject or cascade per schema) and ensure audit logs (PointsLog) are created for points-related CRUD. Automate these with model-level unit tests, Django REST framework API tests (if applicable), and a few end-to-end browser tests for admin UI workflows.

10.4 Error handling & resilience

Confirm the application fails gracefully and returns appropriate HTTP status codes and messages for server and client errors. Test 4xx client errors (bad requests, forbidden, not found) surfacing friendly messages rather than raw tracebacks; test 5xx server errors by simulating downstream failures (database down, object storage unavailable) and ensure the app returns a stable error page and logs the issue without exposing stack traces. Validate timeouts and retries for long-running tasks (e.g., audio transcoding): web endpoints should remain responsive, offload processing to background workers, and provide user-facing progress/queued-state messaging. Test network edge cases such as partial uploads, interrupted connections, and slow clients; file integrity and partial-write protections should prevent corrupted media. Finally, include observability tests: verify that error events are captured by Sentry/logging, health-check endpoints report accurate state, and alerting triggers when key metrics (error rate, queue length, disk usage) cross thresholds. Implement automated smoke tests in CI to detect high-severity failures early and include load or chaos tests for resilience where feasible.

11. Results

The Interview Portal demonstrates significant success in integrating all major components of interview preparation—Aptitude, Technical, and HR—into a single, user-friendly system. One of the strongest outcomes is the platform's ability to simulate real-world interview scenarios through timed quizzes, structured question banks, and audio-based HR practice. These features ensure students gain hands-on experience with the types of assessments they will encounter during campus placements and competitive recruitment processes. The system accurately evaluates user responses, records attempt histories, and stores HR recordings, ensuring that each preparation session contributes meaningfully to the user's learning journey.

Another major result is the effective implementation of performance analytics. The portal generates detailed insight through visual charts that show accuracy trends, topic-wise strengths, time consumption, and progress over multiple attempts. These analytical tools help users clearly identify weak areas and take corrective actions. By presenting quantitative data in an easy-to-understand format, the system enhances self-awareness and encourages targeted practice. This represents a major improvement over traditional test-preparation approaches, which often lack continuous performance tracking.

The gamification engine also contributes significantly to the overall success of the project. Features such as streak counters, points accumulation, and badges for achievements increase user motivation and engagement. This ensures that preparation becomes a consistent and enjoyable routine rather than a stressful, one-time effort. The results show that the platform not only supports assessment and practice but also builds habits, encourages regular usage, and enhances confidence—ultimately fulfilling its goal of providing a holistic interview preparation experience.

12. Conclusion

In conclusion, the Interview Portal delivers a complete and effective solution for students preparing for competitive interviews. By combining MCQ-based assessment, HR audio practice, analytics dashboards, and gamified motivation tools, the system addresses both the technical and interpersonal aspects of interview readiness. The platform's structured design ensures that students receive balanced exposure to different question types and gain the confidence required to perform well in actual interview rounds. Its simplicity, responsiveness, and secure implementation make it accessible to a wide range of learners.

The project also succeeds in streamlining the preparation process by centralizing essential tools into one platform. Traditional approaches to interview preparation often involve using multiple resources, which can be time-consuming and unorganized. The Interview Portal eliminates this inefficiency by offering a consolidated environment where learning, practice, performance assessment, and revision happen seamlessly. This not only saves time but also fosters consistent improvement through structured feedback and analytics.

Overall, the system proves to be highly useful for both academic settings and real-world placement preparation. Its modular architecture allows easy expansion, making it adaptable for future requirements like AI-based evaluation, coding rounds, or proctored tests. The project demonstrates strong technical execution, thoughtful user-centric design, and practical relevance—resulting in a valuable tool that significantly enhances the interview preparation journey for students.

13. Future Enhancements

- AI-Based HR Feedback: Integrate speech-to-text, sentiment analysis, and tone evaluation to generate automated feedback on communication skills, clarity, confidence, and articulation during HR audio recordings.
- Coding Round Integration: Add a built-in code editor with compiler support, test-case evaluation, and scoring for programming questions, enabling preparation for coding interviews along with MCQs and HR rounds.
- Proctoring and Monitoring Features: Implement webcam-based proctoring, face detection, eye-gaze tracking, and tab-switch detection to simulate real online test environments used by modern companies.

- Mobile App / PWA Support: Develop a dedicated mobile application or convert the system into a Progressive Web App (PWA) to allow students to practice quizzes, view analytics, and access HR modules from any device.
- Leaderboard & Social Learning: Introduce leaderboards, weekly challenges, peer comparison analytics, and collaborative preparation spaces to improve user engagement and promote healthy competition.

14. References

1. Django Software Foundation. *Django Documentation*. Available at: https://docs.djangoproject.com
2. Tailwind Labs. *Tailwind CSS Documentation*. Available at: https://tailwindcss.com
3. Chart.js Contributors. *Chart.js Official Documentation*. Available at: https://www.chartjs.org
4. Mozilla Developer Network. *Web Audio API Reference*.
5. SQLite Consortium. *SQLite Technical Documentation*. Available at: https://sqlite.org
6. Malnad College of Engineering. *Django Macro Project Report Format (23IS553)* — provided file: /mnt/data/Django_Project_Report_Format_23IS553.pdf
7. DigitalOcean Docs. *Deploying Django Apps*.
8. PostgreSQL Global Development Group. *PostgreSQL Documentation*.