Django Dashboard: Interview Intelligence Visualization

MCA Minor Project

January 2024

Contents

1 Introduction

1.1 Dashboard Purpose

The Django dashboard serves as the primary interface for viewing and understanding interview intelligence results. It transforms complex processed data into clear, interpretable visualizations that support academic evaluation and decision-making processes. The dashboard maintains strict separation from inference logic, operating purely as a presentation layer for backend-processed data.

1.2 Design Philosophy

The dashboard follows established design principles:

- Read-only access to ensure evaluation integrity
- Clear visual hierarchy for information comprehension
- Minimal interaction to prevent accidental data modification
- Academic-friendly presentation suitable for evaluation contexts
- Progressive disclosure of information complexity

1.3 System Boundaries

The dashboard is specifically responsible for:

- Displaying processed interview data in human-readable formats
- Providing navigation between interview sessions and analysis views
- Visualizing temporal analysis and candidate profiles
- Offering minimal administrative controls for session management

The dashboard explicitly does not handle:

- Media capture or real-time processing
- Live data updates during interview sessions
- Evaluation logic or scoring computations
- Model training, configuration, or parameter adjustment

1.4 Technical Constraints

- Browser compatibility: Chrome 90+, Firefox 88+, Safari 14+
- Session limit: 100 concurrent users maximum
- Data refresh rate: Manual refresh only, no real-time updates
- File upload size: Maximum 10MB for JD uploads

2 Django Architecture

2.1 Application Structure

The dashboard implemented as a Django application:

```
dashboard/
--- models.py          # Read-only models for data access
--- views.py           # View logic for page rendering
--- urls.py            # URL routing configuration
--- serializers.py     # API data serialization
--- templates/         # HTML templates
-   --- dashboard/
-       --- base.html
-       --- sessions.html
-       --- script.html
-       --- qa.html
-       --- profile.html
-       --- verdict.html
--- static/           # CSS and JavaScript assets
-   --- css/
-   --- js/
-   --- images/
--- api.py            # REST API endpoints
--- utils.py          # Helper functions for data processing
```

2.2 Data Access Patterns

The dashboard utilizes optimized database query strategies:

- `select_related()` for foreign key relationship optimization
- `prefetch_related()` for many-to-many relationship optimization
- Database indexing on frequently queried fields (session_id, timestamps)
- Query result caching with 15-minute expiration for expensive aggregations
- Connection pooling with maximum 20 connections

2.3 Security Implementation

Access control and security measures include:

- Session-based authentication with 30-minute timeout
- Permission-based view restrictions using Django's auth system
- SQL injection prevention through ORM usage exclusively
- XSS protection via Django template auto-escaping
- CSRF token validation for all form submissions
- HTTPS enforcement for all dashboard endpoints

3 Data Models Consumption

3.1 Primary Data Sources

The dashboard consumes read-only versions of core Django models:

- `InterviewSession`: Session metadata and overall status
- `ScriptTurn`: Conversation timeline data with speaker attribution
- `QuestionAnswer`: Extracted question-answer pairs with confidence
- `CandidateProfileSnapshot`: Profile evolution data over time
- `FinalVerdict`: Final evaluation results with uncertainty quantification

3.2 Data Aggregation Patterns

Views perform necessary data aggregation operations:

- Profile trend calculations using sliding window averages
- Verdict summary statistics with confidence intervals
- Session status aggregation for overview displays
- Performance metric aggregations with outlier detection
- Temporal aggregation for time-series visualization

3.3 Data Freshness Management

- Data timestamp validation to ensure freshness
- Automatic cache invalidation on data updates
- Manual refresh indicators for stale data warnings
- Background task monitoring for processing status
- Data versioning for audit trail maintenance

4 Page-by-Page Implementation

4.1 Sessions List

4.1.1 Purpose and Functionality

The sessions list serves as the primary entry point for interview evaluation, providing a comprehensive overview of all available interview sessions and their current processing status.

4.1.2 URL Structure

`/dashboard/sessions/`

4.1.3 Data Display Components

The sessions list displays the following information:

- Session identifier (clickable link to detailed view)
- Interview date and time with timezone information
- Processing status (Processing, Completed, Error, Archived)
- Final hire probability when available (percentage with color coding)
- Duration of interview in minutes and seconds
- Number of question-answer pairs successfully extracted
- JD alignment score with visual indicator

4.1.4 Interactive Features

- Sort by date (ascending/descending)
- Sort by score (highest/lowest)
- Filter by status (All, Completed, Processing, Error)
- Search by session ID or candidate identifier
- Pagination with 25 sessions per page
- Bulk operations for multiple session selection

4.1.5 Table Structure

Table 1: Sessions Overview Table Structure

Session ID	**Date**	**Status**	**Score**	**Duration**	**Actions**
session_001	2024-01-15	Completed	63%	45:32	View, Export
session_002	2024-01-16	Processing	-	-	Status
session_003	2024-01-16	Error	-	38:15	Retry, View Log
session_004	2024-01-17	Completed	71%	52:08	View, Export

4.2 Script Timeline

4.2.1 Purpose and Scope

The script timeline displays the complete conversation in chronological order, showing exactly who spoke what and when, providing essential temporal context for the entire interview.

4.2.2 URL Structure

```
/dashboard/session/<id>/script/
```

4.2.3 Display Format Specifications

Conversation displayed with standardized format:

- Timestamp format: [MM:SS] Speaker: Content
- Color coding for speakers (blue for interviewer, green for candidate)
- Scrollable interface for conversations longer than 50 turns
- Click-to-jump navigation to specific timestamps
- Search functionality for keywords within conversation

4.2.4 Example Display Format

```
[00:05] Interviewer: Can you explain your experience with distributed systems?
[00:09] Candidate: I worked on distributed systems for three years, primarily
         focusing on microservices architecture and container orchestration.
[00:25] Interviewer: What specific challenges did you face with microservices?
[00:28] Candidate: The main challenge was handling network partitions and ensuring
         consistency across distributed transactions. We implemented event sourcing
         and CQRS patterns to address these issues.
[00:45] Interviewer: How did you handle service discovery?
```

4.2.5 Navigation and Interaction Features

- Timeline progress indicator showing current position
- Click-to-jump timestamp navigation with smooth scrolling
- Speaker highlighting on hover for improved readability
- Export to text file functionality with formatting options
- Word search with result highlighting
- Turn filtering by speaker

4.3 Question-Answer Analysis

4.3.1 Purpose and Objectives

This page displays extracted question-answer pairs with associated confidence scores, allowing evaluation of the conversation structure and analysis quality.

4.3.2 URL Structure

```
/dashboard/session/<id>/qa/
```

4.3.3 Data Organization and Display

Q/A pairs displayed in chronological order with comprehensive metadata:

- Question text highlighted and indented
- Answer text following each question with proper attribution
- Confidence score displayed as percentage with color coding
- Semantic similarity indicator between question and answer
- Temporal gap information showing response latency
- Question type classification (technical, behavioral, situational)

4.3.4 Display Example with Metadata

```
Q1: Can you explain your experience with distributed systems? [Technical]
A1: I worked on distributed systems for three years, focusing on microservices
    architecture, container orchestration, and service mesh implementations.
Confidence: 82%
Semantic Similarity: 0.78
Response Time: 4.2 seconds
Word Count: Q=9, A=23

Q2: What specific technologies did you use for service discovery? [Technical]
A2: We used Consul for service discovery with health checking, integrated with
    Kubernetes service discovery for cloud-native deployments.
Confidence: 75%
Semantic Similarity: 0.81
Response Time: 2.1 seconds
Word Count: Q=9, A=18
```

4.3.5 Quality Indicators and Visualization

- Color-coded confidence levels (green ¿ 80%, yellow 60-80%, red ¡ 60%)
- Missing answer indicators with analysis explanations
- Question classification labels with visual icons
- Answer length and complexity metrics displayed graphically
- Response time distribution across all Q/A pairs

4.4 Profile Evolution

4.4.1 Purpose and Rationale

The profile evolution page shows how candidate evaluation metrics change over time, demonstrating the incremental and adaptive nature of the assessment system.

4.4.2 URL Structure

`/dashboard/session/<id>/profile/`

4.4.3 Profile Dimensions and Metrics

Five core dimensions tracked throughout the interview:

1. **Technical Depth**: Domain knowledge and expertise level assessment
2. **JD Relevance**: Alignment with job requirements analysis
3. **Clarity**: Communication effectiveness evaluation
4. **Consistency**: Logical coherence measurement across answers
5. **Confidence**: Linguistic certainty and assertiveness indicators

4.4.4 Tabular Data Presentation

Table 2: Candidate Profile Evolution Over Time

Time	**Technical**	**JD Relevance**	**Clarity**	**Consistency**	**Confidence**
120s	0.62	0.58	0.64	0.59	0.61
240s	0.68	0.65	0.66	0.63	0.65
360s	0.71	0.69	0.68	0.67	0.70
480s	0.73	0.72	0.71	0.70	0.72
600s	0.74	0.74	0.73	0.72	0.74

4.4.5 Visual Elements and Charts

- Simple line charts showing trends for each dimension
- Color-coded improvement indicators (green for positive trends)
- Standard deviation bands showing stability over time
- Annotation markers for significant events or changes
- Comparative overlays showing average profiles for reference

4.4.6 Statistical Summary Section

- Average scores per dimension with confidence intervals
- Final vs. initial score comparison with percentage change
- Stability metrics (variance and coefficient of variation)
- Improvement rate calculations with statistical significance
- Correlation analysis between different dimensions

4.5 Final Verdict

4.5.1 Purpose and Significance

The verdict page provides a comprehensive summary of the final evaluation, including hire probability, risk factors, and supporting evidence with uncertainty quantification.

4.5.2 URL Structure

```
/dashboard/session/<id>/verdict/
```

4.5.3 Primary Evaluation Metrics

Core assessment metrics displayed prominently:

- Hire Probability: Overall assessment score (0-100% scale)
- Confidence Level: Uncertainty quantification of the verdict
- JD Alignment Score: Specific job requirement matching
- Final Composite Score: Weighted combination of all factors
- Risk Flag Summary: Identified concerns and warnings

4.5.4 Comprehensive Display Format

```
Overall Assessment
-----------------
Hire Probability: 63%
Confidence Level: Medium (58%)
JD Alignment Score: 71%
Final Composite Score: 66%
Confidence Interval: [51%, 75%]

Evidence Breakdown
------------------
Q/A Quality: 68% (Weight: 40%)
Profile Stability: 72% (Weight: 30%)
Behavioral Consistency: 51% (Weight: 20%)
Technical Accuracy: 74% (Weight: 10%)

Risk Factors Analysis
-------------------
[!] Overgeneralization detected in responses (Confidence: 82%)
[!] Inconsistent skill level descriptions across answers (Confidence: 75%)
[!] Limited concrete examples provided for complex topics (Confidence: 68%)

Statistical Validation
---------------------
Statistical Significance: p < 0.05
Model Calibration: Brier Score = 0.18
```

```
Cross-Validation Accuracy: 74.2%
Expert Agreement: 78% concordance

Recommendation
--------------
Proceed with caution - candidate shows potential but requires
additional technical evaluation for high-stakes positions. Consider
technical interview with senior team members for final assessment.
```

4.5.5 Supporting Details and Evidence

- Evidence weight distribution with rationale
- Risk factor explanations with supporting quotes
- Comparison with historical averages and benchmarks
- Recommended next steps and additional evaluations
- Data quality indicators and processing completeness

5 API Integration and Data Access

5.1 Internal API Consumption

The dashboard consumes internal REST APIs from processing backend:

- Session data retrieval with comprehensive metadata
- Script and Q/A data with pagination for large datasets
- Profile evolution data with time-series formatting
- Verdict data with uncertainty quantification
- Processing status and progress information

5.2 Data Caching Strategy

Intelligent caching implementation for performance:

- Session overview data cached for 5 minutes
- Script and Q/A data cached for 15 minutes
- Profile evolution data cached for 10 minutes
- Verdict data cached for 30 minutes
- Automatic cache invalidation on data updates

5.3 Error Handling and Fallbacks

- Graceful degradation when APIs unavailable
- Cached data fallback for temporary service issues
- User-friendly error messages with retry options
- Automatic retry with exponential backoff for failed requests
- Logging and monitoring for error tracking

6 Read-Only Guarantees Implementation

6.1 Data Integrity Protection

The dashboard maintains read-only access through multiple layers:

- Read-only Django model managers with save methods disabled
- Form validation disabled for display-only purposes
- Database transactions limited to read operations only
- API endpoints restricted to GET methods exclusively
- Write attempt detection with logging and blocking

6.2 Prevention of Data Modification

Multiple mechanisms prevent accidental data changes:

- No edit forms provided for evaluation data
- Disabled form fields in templates where editing might be expected
- Client-side validation preventing form submissions
- Server-side permission checks blocking write operations
- Audit trail logging for any attempted modifications

6.3 Audit Trail and Accountability

All dashboard access logged for comprehensive accountability:

- User session tracking with IP and timestamp
- Page view timestamps and durations
- Data access patterns and frequency analysis
- Export operation logging with user attribution
- Failed access attempt monitoring and alerting

7 Manual Controls and Administrative Features

7.1 Profiling Control Options

Limited administrative controls available for authorized users:

- Toggle profiling ON/OFF for active interview sessions
- Mark interview as completed for archival purposes
- Add examiner comments stored separately from evaluation data
- Override processing status for troubleshooting

7.2 Session Management Capabilities

- Retry failed processing stages with parameter adjustment
- Archive completed sessions to improve performance
- Export session data to CSV/JSON with customizable fields
- Bulk operations for multiple session management
- Session deletion with confirmation and audit logging

7.3 System Monitoring Features

- View system processing status and queue lengths
- Monitor active processing jobs with progress indicators
- Access system logs for debugging and analysis
- Performance metrics dashboard with historical trends
- Resource utilization monitoring and alerting

8 Performance Optimization

8.1 Database Optimization Strategies

- Strategic indexing on session_id, timestamps, and status fields
- Query result pagination with default 25 items per page
- Lazy loading for expensive computational results
- Database connection pooling with 20 connection maximum
- Query optimization with Django Debug Toolbar monitoring

8.2 Caching Implementation

- Redis backend for session-based data caching
- Template fragment caching for static components
- Browser caching optimization for static assets
- CDN deployment configuration for production environments
- Cache warming strategies for frequently accessed data

8.3 Asset Optimization

- CSS minification and bundling for faster loading
- JavaScript compression with source map generation
- Image optimization for charts and visualization elements
- HTTP/2 server push for critical resources
- Asset versioning for cache busting on updates

9 Security Implementation Details

9.1 Access Control Mechanisms

- Role-based permissions (examiner, administrator, viewer)
- Session timeout after 30 minutes of inactivity
- Password protection for sensitive evaluation data
- IP whitelisting configuration for production deployment
- Multi-factor authentication requirement for administrators

9.2 Data Privacy Protection

- Candidate data anonymization options for compliance
- Secure transmission with HTTPS enforced everywhere
- Data retention policies with automatic cleanup
- Right to data deletion compliance implementation
- GDPR-compliant data handling procedures

9.3 Input Validation and Sanitization

- URL parameter validation with type checking
- SQL injection prevention through ORM exclusively
- XSS protection in template rendering with auto-escaping
- CSRF token validation for all form submissions
- File upload validation for JD documents

10 Implementation Roadmap and Testing

10.1 Development Build Sequence

Systematic development approach with clear milestones:

1. Django project setup and base configuration
2. Database models design and migrations implementation
3. Basic template structure and CSS framework integration
4. Sessions list page implementation with pagination
5. Script timeline view development with search functionality
6. Q/A analysis page creation with confidence visualization
7. Profile evolution visualization with charting
8. Verdict summary page with comprehensive metrics
9. API integration and error handling implementation
10. Performance optimization and caching strategies

10.2 Testing Strategy Implementation

Comprehensive testing across multiple dimensions:

- Unit tests for view logic and helper functions
- Integration tests for API consumption and data flow
- Frontend testing for user interactions and navigation
- Performance testing with large datasets and concurrent users
- Security testing for vulnerability assessment and penetration testing
- Accessibility testing for compliance with WCAG 2.1 standards

11 Conclusion

The Django dashboard provides a comprehensive and academically appropriate interface for interview intelligence visualization. By maintaining strict separation from processing logic and focusing on clear presentation of evaluation results, the dashboard supports effective academic assessment and decision-making processes. The modular design allows for independent development and testing while ensuring data integrity and performance optimization.

The emphasis on read-only access, comprehensive error handling, and security implementation ensures that evaluation data remains trustworthy and tamper-proof. The intuitive interface design with progressive information disclosure enables users at various technical levels to effectively utilize the system for interview assessment and research purposes. The dashboard successfully transforms complex ML-driven evaluation results into actionable, interpretable insights suitable for academic and professional applications.