Software Design Specifications

[CloutCheck AI]

Version: [01.0]

Project Code	F25-220
Supervisor	Sir Basit Ali Jasani
Co Supervisor	Dr. Muhammad Rafi
Project Team	Syed Uzair Hussain - 22k4212 Saad Ahmed - 22k4345 Huzaifa Bin Khalid - 22k4223
Submission Date	4-12-2025

[Instructions]

- *No section of template should be deleted. You can write 'Not applicable' if a section is not applicable to your project. But all sections must exist in the final document.*
- *All comments/examples mentioned in square brackets ([]) are in the template for explanation purposes and must be replaced / removed in final document.*
- *This' Instruction' section should also be removed in final document.*
- *MS-Word Reviewing feature must be used to get the document reviewed by PMs or supervisors.*

Document History

[Revision history will be maintained to keep a track of changes done by anyone in the document.]

Version	Name of Person	Date	Description of change
1.0	Syed Uzair Hussain	*27/11/25*	Document Created
1.0	Saad Ahmed	*28/11/25*	Added System Architecture section + improved ER Diagram description
1.0	Huzaifa Bin Khalid	*29/11/25*	Added Design Strategy + expanded Risks & Dependencies
1.0	Saad Ahmed	*1/12/25*	Added Database Data Dictionary + refined tables
1.0	Syed Uzair Hussain	*2/12/25*	Added Application Design + created Sequence Diagrams
1.0	Huzaifa Bin Khalid	*2/12/25*	Added State Diagrams + Report lifecycle flow
1.0	Syed Uzair Hussain	*3/12/25*	Final editing & proofreading

Distribution List

[Following table will contain list of people whom the document will be distributed after every sign-off]

Name	Role
Sir Basit Ali Jasani	Supervisor
Dr. Muhammad Rafi	Co Supervisor

Document Sign-Off

[Following table will contain sign-off details of the document. Once the document is prepared and revised, this should be signed-off by the sign-off authority.
Any subsequent changes in the document after the first sign-off should again get a formal sign-off by the authorities.]

Version	Sign-off Authority	Project Role	Signature	Sign-off Date
1.0	Supervisor	Supervisor	Sir Basit Ali Jasani	2/12/2025

Document Information

Category	Information
Customer	FAST-NU
Project	CloutCheck AI
Document	Software Design Specification
Document Version	1.0
Status	Draft
Author(s)	Syed Uzair Hussain, Saad Ahmed, Huzaifa Bin Khalid
Approver(s)	Sir Basit Ali Jasani
Issue Date	December 2025
Document Location	Project Repository
Distribution	Advisor Project Coordinator’s Office (through Advisor)

Definition of Terms, Acronyms and Abbreviations

[This section should provide the definitions of all terms, acronyms, and abbreviations required to interpret the terms used in the document properly.]

Term	Description
AI	Artificial Intelligence
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
CLIP	Contrastive Language-Image Pre-training
CV	Computer Vision
ML	Machine Learning
NLP	Natural Language Processing
NSFW	Not Safe For Work
RBAC	Role-Based Access Control
ResNet	Residual Neural Network
RoBERTa	Robustly Optimized BERT Pretraining Approach
SaaS	Software as a Service
SDK	Software Development Kit
ViT	Vision Transformer
YOLO	You Only Look Once

Table of Contents

1 Introduction 8

1.1 Purpose of Document 8
1.2 Intended Audience 8
1.3 Document Convention 8
1.4 Project Overview 8
1.5 Scope 8

2 Design Considerations 9

2.1 Assumptions and Dependencies 9
2.2 Risks and Volatile Areas 9

3 System Architecture 10

3.1 System Level Architecture 10
3.2 Software Architecture 10

4 Design Strategy 11

5 Detailed System Design 12

5.1 Database Design 12
5.1.1 ER Diagram 12
5.1.2 Data Dictionary 12
5.1.2.1 User 12
5.1.2.2 Influencer_Profile 12
5.1.2.3 Content_Items 12
5.1.2.4 Text_Content 12
5.1.2.5 Image_Content 12
5.1.2.6 Audio_Content 12
5.1.2.7 Video_Content 12
5.1.2.8 Brand_Profile_Matching (Multimodal Fusion) 12
5.1.2.9 Report 12

5.2 Application Design 14
5.2.1 Sequence Diagram 14
5.2.1.1 Complete Influencer Scan Workflow 14
5.2.1.2 User Authentication and Profile Management 14
5.2.1.3 View Scan Results Dashboard 14
5.2.2 State Diagram 14
5.2.2.1 Scan Job Lifecycle State Diagram 14
5.2.2.2 User Account Lifecycle State Diagram 14
5.2.2.3 Report Generation Lifecycle State Diagram 14

6 Appendices 16

1 Introduction

1.1 Purpose of Document

The purpose of this Software Design Specification (SDS) document is to provide a comprehensive and detailed design blueprint for **CloutCheck AI**, a multimodal reputation risk analysis platform. This document serves as a reference for all stakeholders including developers, testers, project managers, and supervisors to understand:

- The architectural design and component interactions
- Database schema and data flow
- AI/ML model integration strategies
- System interfaces and API specifications
- Security and scalability considerations

This document will guide the implementation phase and serve as the foundation for system testing, validation, and future maintenance activities.

1.2 Intended Audience

- Fast NU
- Jury
- Supervisor
- Co-Supervisor
- Students of Fast NU
- Our Team(Designer, Developer, Tester)
- Potential Users of this product

1.3 Document Convention

- Font Family = Arial
- Font Size = 12 for headings, 10 for the rest of the content

1.4 Project Overview

CloutCheck AI is an intelligent SaaS platform designed to assess reputation risks associated with influencers, public figures, and job candidates by analyzing their publicly available social media content across multiple platforms (YouTube, TikTok, Twitter/X, Instagram).

The system leverages state-of-the-art AI technologies including:

- Natural Language Processing (NLP) for text analysis
- Computer Vision for image and video frame analysis
- Speech Recognition and Audio NLP for audio content analysis
- Multimodal Fusion techniques to combine insights from all modalities

Key Objectives:

- Automated content scraping from social media platforms
- Detection of harmful content (hate speech, violence, NSFW, drugs, extremism)
- Generation of transparent reputation risk scores
- Production of professional, downloadable reports
- Real-time monitoring and alerting capabilities

1.5 Scope

Included Functionalities:

1. **User Management**
   - Secure authentication and authorization
   - Role-based access control (Admin, Brand Manager, HR Professional)
   - User profile management
2. **Content Acquisition**
   - Integration with YouTube, TikTok, Twitter/X, Instagram APIs
   - Custom web scraping for publicly available content
   - Support for manual profile URL submission
3. **Multimodal AI Analysis**
   - Text toxicity and sentiment analysis
   - Image classification for NSFW, violence, drugs
   - Video frame extraction and analysis
   - Speech-to-text transcription and audio toxicity detection
4. **Risk Scoring and Classification**
   - Category-based risk detection (hate speech, violence, NSFW, drugs, extremism, misinformation)
   - Weighted multimodal fusion
   - Overall reputation risk score (0-100 scale)
5. **Dashboard and Reporting**
   - Interactive web-based dashboard
   - Visualization of risk metrics (charts, graphs, heatmaps)
   - Trend analysis over time
   - PDF report generation with branding

2 Design Considerations

[This section describes many of the issues which need to be addressed or resolved before attempting to devise a complete design solution. In other words, this section is used to formally set the groundwork for the system design.]

2.1 Assumptions and Dependencies

Assumptions:

1. **Data Availability**: Social media platforms will continue to provide public API access or allow web scraping within their terms of service
2. **Internet Connectivity**: Users will have stable internet connections for content scraping and analysis
3. **Computational Resources**: Sufficient cloud computing resources (GPU/CPU) will be available for AI model inference
4. **Model Accuracy**: Pre-trained models will achieve baseline accuracy sufficient for MVP deployment
5. **User Expertise**: Users have basic understanding of social media platforms and reputation management concepts
6. **Data Volume**: Initial user base will generate manageable data volumes within free/student-tier cloud limits

Dependencies:

1. **External APIs**:
    - Apify API (for scraping)
2. **AI/ML Frameworks**:
    - PyTorch or TensorFlow for model training
    - Hugging Face Transformers library
    - OpenCV for video processing
    - Whisper for speech recognition
3. **Infrastructure**:
    - AWS, Google Cloud Platform, or Azure for hosting
    - Docker for containerization
    - MongoDB for data storage
    - FastAPI for backend services
4. **Development Tools**:
    - Git/GitHub for version control
    - React.js for frontend development
    - Postman for API testing

2.2 Risks and Volatile Areas

Technical Risks:

1. **API Rate Limiting**: Social media platforms may impose strict rate limits, affecting scraping speed
    - Mitigation: Implement request queuing, caching, and fallback to web scraping
2. **Model Inference Latency**: AI models may be too slow for real-time analysis
    - Mitigation: Use model quantization, batch processing, and asynchronous job queues
3. **False Positives/Negatives**: AI models may misclassify content
    - Mitigation: Ensemble methods, confidence thresholds, human-in-the-loop validation
4. **Scalability**: System may not handle high concurrent users
    - Mitigation: Horizontal scaling, load balancing, caching strategies

Compliance and Legal Risks:

1. **Terms of Service Violations**: Scraping may violate platform ToS
    - Mitigation: Prioritize official APIs, implement respectful scraping practices, legal consultation
2. **Privacy Concerns**: Analysis of public content may raise privacy issues
    - Mitigation: Only analyze publicly available content, clear user consent, privacy policy
3. **Bias and Fairness**: AI models may exhibit bias against certain demographics
    - Mitigation: Bias auditing, explainability features

Operational Risks:

3 **Cost Overruns**: Cloud and API costs may exceed budget. Mitigation: Cost monitoring, optimization, free-tier maximization

4 **Team Availability**: Team members may have conflicting academic commitments Mitigation: Clear task allocation, regular check-ins, contingency planning

3 System Architecture

[This section should provide a high-level overview of how the functionality and responsibilities of the system are partitioned and then assigned to subsystems or components. The main purpose is to gain a general understanding of how the system is decomposed, and how the individual parts work together to provide the desired functionality].

3.1 System Level Architecture

The CloutCheck AI system follows a microservices-oriented architecture with clear separation of concerns across five primary layers:

Layer 1: Data Ingestion & Preprocessing

- Responsible for extracting content from social media platforms
- Handles data cleaning, normalization, and storage
- Components: API Integrators, Web Scrapers, Data Validators

Layer 2: Feature Extraction & Processing

- Processes raw content into AI-ready formats
- Separate pipelines for text, image, audio, and video
- Components: NLP Preprocessors, Image Processors, Audio Transcribers, Video Frame Extractors

Layer 3: AI/ML Model Inference

- Hosts specialized AI models for each modality
- Performs risk classification and confidence scoring
- Components: Text Models (BERT/RoBERTa), Vision Models (YOLO/ResNet), Audio Models (Whisper + NLP)

Layer 4: Multimodal Fusion & Scoring

- Combines insights from all modalities
- Applies fusion strategies and generates unified risk scores
- Components: Fusion Engine, Risk Classifier, Score Calculator

Layer 5: Application & Presentation

- User-facing interfaces and report generation
- Backend APIs for orchestration
- Components: Dashboard (React), Backend APIs (FastAPI), Report Generator

Cross-Cutting Concerns:

5 **Security Layer**: Authentication, authorization, encryption
6 **Data Layer**: MongoDB for metadata, S3 for media files
7 **Monitoring Layer**: Logging, metrics, alerting

3.2 Software Architecture

The software architecture follows a three-tier model:

Presentation Tier (Frontend)

- **Technology**: React.js with TypeScript
- **Components**:
  - Authentication Module (Login, Registration, Password Reset)
  - Dashboard Module (Risk Scores, Visualizations, Trends)
  - Scan Management Module (Initiate Scans, View History)
  - Report Module (Generate, Download PDF Reports)
  - Settings Module (User Preferences, API Keys)
- **Communication**: RESTful API calls to backend
- **Styling**: Tailwind CSS or Material-UI

Application Tier (Backend)

- **Technology**: Python FastAPI
- **Components**:
  - **Authentication Service**: JWT-based auth, RBAC
  - **Scraping Service**: Orchestrates API calls and web scrapers
  - **Processing Service**: Routes content to appropriate AI pipelines
  - **Inference Service**: Manages AI model loading and prediction
  - **Fusion Service**: Combines multimodal results
  - **Reporting Service**: Generates PDF reports
  - **Notification Service**: Sends alerts via email/SMS
- **Communication**:
  - Frontend ↔ Backend: REST APIs (JSON)
  - Backend ↔ AI Models: gRPC or direct Python calls
  - Backend ↔ Database: MongoDB driver
- **Job Queue**: Celery with Redis for asynchronous task processing

Data Tier (Storage)

- **Primary Database**: MongoDB
  - Stores user accounts, scan metadata, risk scores
  - Collections: Users, Scans, Results, Reports
- **File Storage**: AWS S3 or Google Cloud Storage
  - Stores downloaded media files (images, videos, audio)
  - Stores generated PDF reports
- **Cache**: Redis

    - Caches API responses, frequently accessed data
    - Stores session information
- **Model Storage**: Model registry or local filesystem
    - Stores trained AI model weights
    - Version control for models

4 Design Strategy

The design strategy for CloutCheck AI emphasizes modularity, scalability, maintainability, and ethical responsibility. Key principles include:

1. Modular Component Design

- Each AI pipeline (text, image, audio, video) operates independently
- Enables parallel development and testing
- Facilitates model swapping and upgrades without system-wide changes
- Supports addition of new platforms or risk categories

2. API-First Development

- All backend functionality exposed via RESTful APIs

3. Asynchronous Processing

- Long-running tasks (scraping, AI inference) handled asynchronously
- Improves user experience with non-blocking operations
- Enables horizontal scaling of worker processes

4. Scalability Considerations

- Stateless backend services for horizontal scaling
- Load balancing across multiple API instances
- Database sharding strategy for future growth
- CDN usage for static assets

5. Explainability and Transparency

- AI model outputs include confidence scores
- Visual attention maps for image/video classifications
- Keyword/phrase highlighting for text toxicity
- Clear documentation of scoring methodology

6. Ethical Safeguards

- Bias auditing of AI models across demographics
- Fairness metrics evaluation
- User consent mechanisms
- Data retention and deletion policies
- Compliance with GDPR-like standard

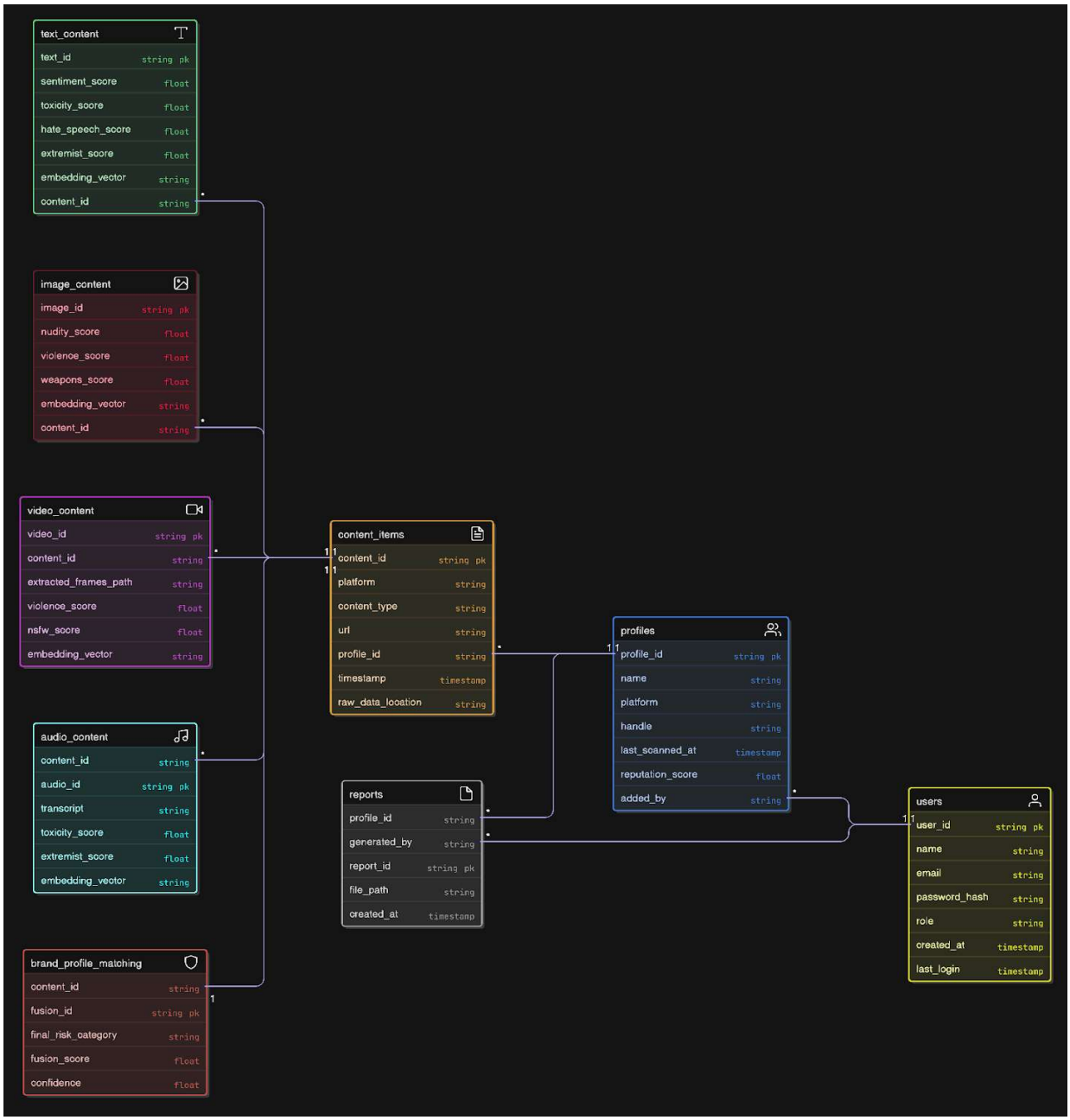
7. Cost Optimization

- Efficient use of cloud resources (auto-scaling)
- Batch processing to reduce API calls
- Model quantization for faster inference.

5 Detailed System Design

5.1 Database Design

5.1.1 ER Diagram



5.1.2 Data Dictionary

5.1.2.1. User

Name	The user registered in the system to access the software.
Where-used/how-used	Stores all registered users of the system (Admins, Brand Managers, HR, Parents, Analysts). Used for authentication, role-based authorization, and mapping who added profiles or generated reports.
Content description	Contains key personal and system-level metadata related to authentication, access control, and activity tracking.

Column Name	Description	Type	Length	Nullable	Default Value	Key Type
user_id	Unique identifier of the user	INT	11	NO	None	PK
name	Full name of the user	VARCHAR	150	NO	None	
email	Login email address	VARCHAR	150	NO	None	UNIQUE
password_hash	Encrypted password	VARCHAR	255	NO	None	
role	User type/permission group	VARCHAR	50	NO	'manager'	
created_at	Account creation timestamp	DATETIME	8	NO	CURRENT_TIMESTAMP	
last_login	Most recent login time	DATETIME	8	YES	NULL	

5.1.2.2 Influencer_Profile

Name	The influencer profile stored/added in the database
Where-used/how-used	Represents influencers or public figures whose social media content is analyzed. Used by the scraping engine, risk scoring module, and reporting system.
Content description	Contains metadata about influencer accounts including handle, platform, and reputation score.

Column Name	Description	Type	Length	Nullable	Default Value	Key Type
profile_id	Unique identifier for the profile	INT	11	NO	None	PK
name	Name of influencer/person	VARCHAR	150	NO	None	
platform	Social platform source	VARCHAR	50	NO	None	
handle	Username or account link	VARCHAR	150	NO	None	
added_by	User who added this profile	INT	11	NO	None	FK
last_scanned_at	Timestamp of last analysis	DATETIME	8	YES	NULL	
reputation_score	Final computed reputation score (0-100)	FLOAT	8	YES	NULL	

5.1.2.3 Content_Items

Name	The content stored of the influencer.

Where-used/how-used	Used by the scraping engine, AI analysis modules, and risk scoring components. Represents all multimodal content fetched from social platforms.
Content description	Stores each piece of content (text, image, audio, video) along with metadata and storage path.

Column Name	Description	Type	Length	Nullable	Default Value	Key Type
content_id	Unique ID for content	INT	11	NO	None	PK
profile_id	Linked influencer profile	INT	11	NO	None	FK
platform	Platform source	VARCHAR	50	NO	None	
content_type	Type of content (text/image/video/audio)	VARCHAR	20	NO	None	
url	URL of original post	TEXT	—	NO	None	
timestamp	Post published time	DATETIME	8	YES	NULL	
raw_data_location	File storage/bucket path	TEXT	—	YES	NULL	

5.1.2.4 Text_Content

Name	The text content of the influencer.
Where-used/how-used	Used by the NLP engine to classify sentiment, toxicity, hate speech, extremist content..
Content description	Stores text analysis results.

Column Name	Description	Type	Length	Nullable	Default Value	Key Type
text_id	Unique text analysis ID	INT	11	NO	None	PK
content_id	Linked content item	INT	11	NO	None	FK
sentiment_score	Sentiment output	FLOAT	8	YES	NULL	
toxicity_score	Toxic language score	FLOAT	8	YES	NULL	
hate_speech_score	Hate speech detection score	FLOAT	8	YES	NULL	
extremist_score	Extremism probability	FLOAT	8	YES	NULL	

5.1.2.5 Image_Content

Name	The image media of the influencer.
Where-used/how-used	Computer vision module uses it to detect nudity, drugs, weapons, and violence.
Content description	Stores CV results and image analysis results

Column Name	Description	Type	Length	Nullable	Default Value	Key Type
image_id	Unique image analysis ID	INT	11	NO	None	PK
content_id	Linked content item	INT	11	NO	None	FK
nudity_score	NSFW probability	FLOAT	8	YES	NULL	
violence_score	Violence detection	FLOAT	8	YES	NULL	
weapons_score	Weapon-related detection	FLOAT	8	YES	NULL	

drugs_score	Drugs-related detection	FLOAT	8	YES	NULL	

5.1.2.6 Video_Content

Name	The video content of the influencer.
Where-used/how-used	Used for frame extraction and harmful content detection.
Content description	Stores analysis results.

Column Name	Description	Type	Length	Nullable	Default Value	Key Type
video_id	Video analysis ID	INT	11	NO	None	PK
content_id	Related content item	INT	11	NO	None	FK
extracted_frames_path	Storage path for frames	TEXT	—	YES	NULL	
violence_score	Frame-level violence score	FLOAT	8	YES	NULL	
nsfw_score	NSFW probability	FLOAT	8	YES	NULL	

5.1.2.7 Audio_Content

Name	The transcribed audio from the video content
Where-used/how-used	Extracted transcripts and toxicity analysis of spoken content.
Content description	Stores Whisper transcripts and audio-derived toxicity scores.

Column Name	Description	Type	Length	Nullable	Default Value	Key Type
audio_id	Audio analysis ID	INT	11	NO	None	PK
content_id	Related content item	INT	11	NO	None	FK
transcript	Extracted text from speech	TEXT	—	YES	NULL	
toxicity_score	Toxic language score	FLOAT	8	YES	NULL	
extremist_score	Extremist content probability	FLOAT	8	YES	NULL	

5.1.2.8 Brand_Profile_Matching (Multimodal Fusion)

Name	The score of how well does the influencer match with a specific brand niche.
Where-used/how-used	Final multimodal fusion results used for risk scoring and reporting.
Content description	Represents the unified risk category, score, and model confidence for each content item.

Column Name	Description	Type	Length	Nullable	Default Value	Key Type
fusion_id	Fusion record ID	INT	11	NO	None	PK
content_id	Linked content	INT	11	NO	None	FK
final_risk_category	Final risk classification	VARCHAR	50	NO	None	
fusion_score	Weighted risk score (0–1 or 0–100)	FLOAT	8	NO	None	

confidence	Prediction confidence	FLOAT	8	YES	NULL	

5.1.2.9 Reports

Name	The comprehensive report generated for the influencer
Where-used/how-used	Stores generated risk assessment reports for external stakeholders.
Content description	Each report is tied to a profile and generated by a specific user.

Column Name	Description	Type	Length	Nullable	Default Value	Key Type
report_id	Unique report ID	INT	11	NO	None	PK
profile_id	Profile for whom the report is generated	INT	11	NO	None	FK
generated_by	User who generated the report	INT	11	NO	None	FK
file_path	Storage path of PDF	VARCHAR	255	NO	None	
created_at	Report generation timestamp	DATETIME	8	NO	CURRENT_TIMESTAMP	

5 Application Design

[A detailed application design should include the following:

- *Detailed Sequence diagram with parameter list*
- *State Transition Diagram*
- *DFD level 1 diagram]*

5.2.1 Sequence Diagram

5.2.1.1 Complete Influencer Scan Workflow

This sequence diagram illustrates the end-to-end process of scanning an influencer's social media content, from user initiation through multimodal AI analysis to final reputation score generation. It represents the core functionality of the CloutCheck AI platform.

The workflow begins when a User enters influencer social media handles through the Frontend Dashboard. The system validates the input and sends a scan request to the API Gateway, which checks if the influencer profile already exists in the database. If not, a new profile is created. The API then creates a ScanJob record with a "queued" status and adds it to the Queue Manager for asynchronous processing.

Once queued, the Content Scraper service picks up the job and updates its status to "processing." The scraper then performs parallel content collection from multiple platforms (Instagram, TikTok, Twitter/X), gathering posts, images, videos, and audio content. Each piece of collected content is uploaded to Cloud Storage (S3) for persistence, and corresponding metadata records are saved in the MongoDB database.

After scraping completes, the system initiates multimodal analysis through three parallel processing streams:

Once all modality-specific analyses complete, the Fusion Engine retrieves all RiskAnalysis results from the database and applies a multimodal fusion strategy) to combine insights across modalities. The fused results are aggregated by risk category. The Risk Scoring Service then calculates a weighted reputation score (0-100 scale) based on the severity and frequency of detected risks. Both the ReputationScore and final ScanJob status ("completed") are saved to the database.

Components Involved: 14 major components including frontend, backend services, AI models, database, storage, and notification system.

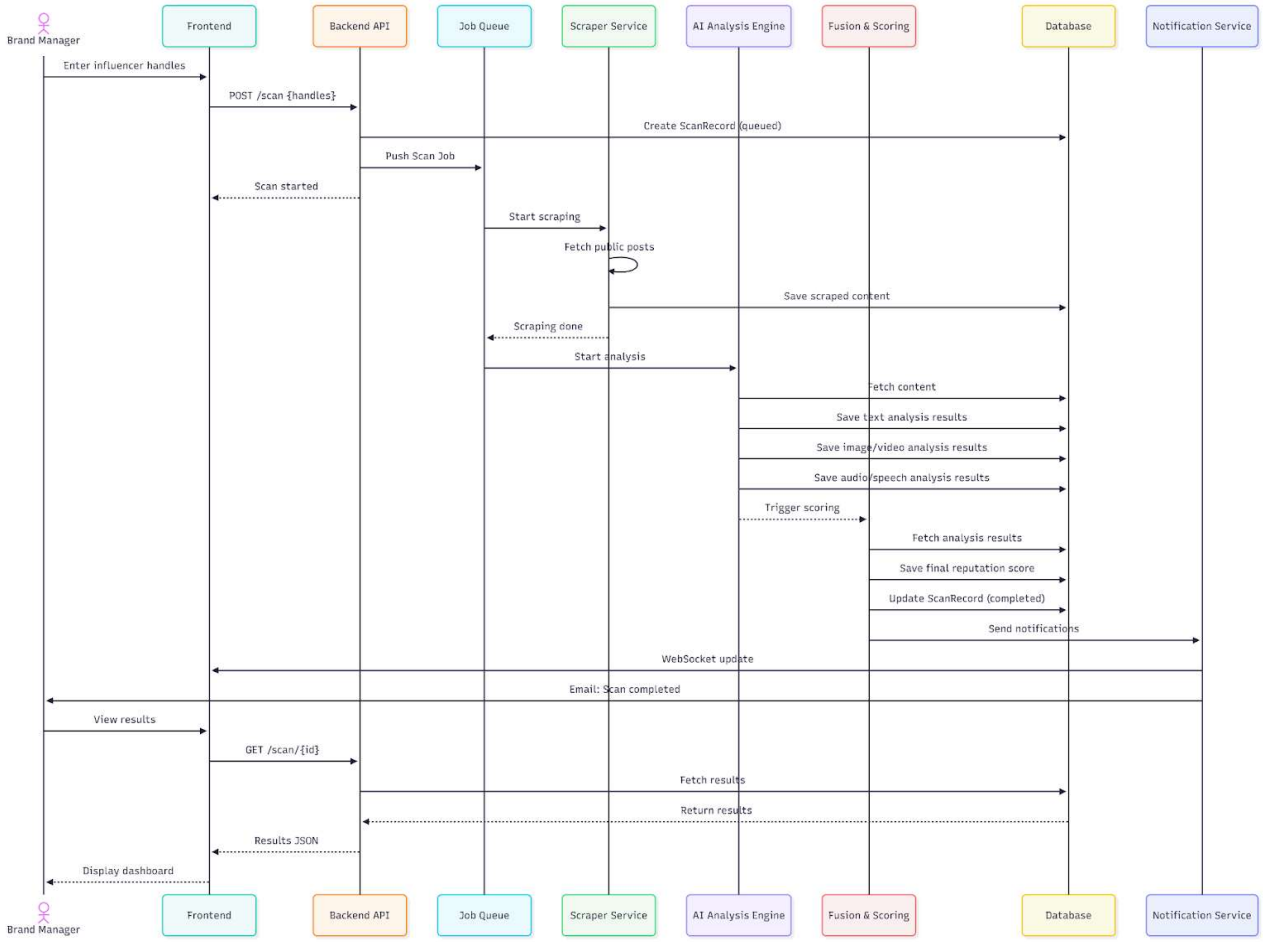


Fig 1.1 Complete Scan Job

5.2.1.2 User Authentication and Profile Management

This sequence diagram demonstrates the security and access control mechanisms of CloutCheck AI, showing both user registration with email verification and the login process with JWT-based authentication. It illustrates how the system ensures only authorized users can access the platform.

Flow A: User Registration

The registration process begins when a user fills out the registration form on the frontend, providing email, password, and role selection (Brand Manager, HR Professional, or Agency User). The frontend performs client-side validation to check email format and password strength requirements.

Upon submission, the registration request is sent to the API Gateway, which forwards it to the Authentication Service. The service first queries the User Database to check if the email already exists. If the email is already registered, a 409 Conflict error is returned, and the user is prompted to login or use a different email.

If the email is available, the system proceeds with account creation. The password is passed to the Password Hasher component, which applies bcrypt hashing with a salt to securely store the credential. The hashed password, along with email, role, and an initial is_active flag set to false, is inserted into the User Database, generating a unique user_id.

The Authentication Service then generates a time-limited verification token via the JWT Token Service and sends a verification email to the user's registered address. The user receives the email, clicks the verification link, which redirects to the frontend with the token as a URL parameter.

The frontend sends a verification request to the API, which validates the token through the JWT service. If valid, the User Database is updated to set is_active to true, activating the account. The user receives confirmation and is directed to the login page.

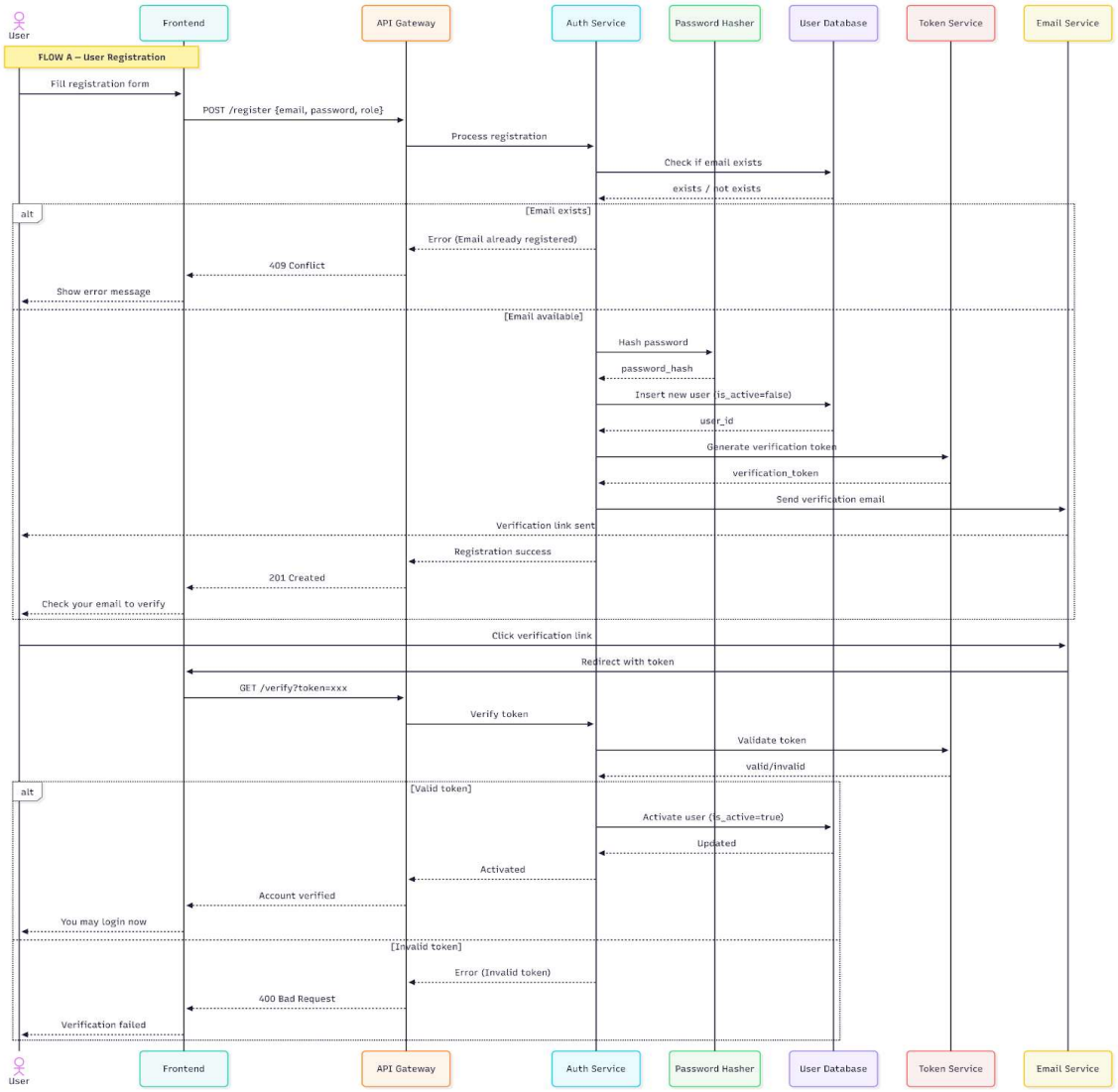


Fig 1.2A Flow A User Registration

Flow B: User Login

The login process starts when a user enters their email and password on the login page. After client-side validation, the credentials are sent to the API Gateway, which forwards them to the Authentication Service.

The service queries the User Database to find a user record matching the provided email. If no user is found, a generic "Invalid credentials" error is returned (to prevent user enumeration attacks). If a user exists, the system checks the is_active status. If the account is not yet verified, a 403 Forbidden response is returned with a message to verify the email.

For active accounts, the Password Hasher compares the provided password against the stored bcrypt hash. If the password is incorrect, another "Invalid credentials" error is returned. This generic messaging prevents attackers from determining whether the email or password was wrong.

When credentials are valid, the JWT Token Service generates an access token containing the user's user_id, role, and expiration time. A session record is created in the database to track active sessions. The last_login timestamp is updated in the user record for audit purposes.

The access token and basic user data (role, email) are returned to the frontend. The UI stores the token in localStorage, sets it in the Authorization header for subsequent API calls, and redirects the user to the main dashboard. All future requests include this token for authorization, which the API Gateway validates before allowing access to protected resources.

Components Involved: 9 components including UI, API, authentication service, database, JWT service, email service, and session manager.

.

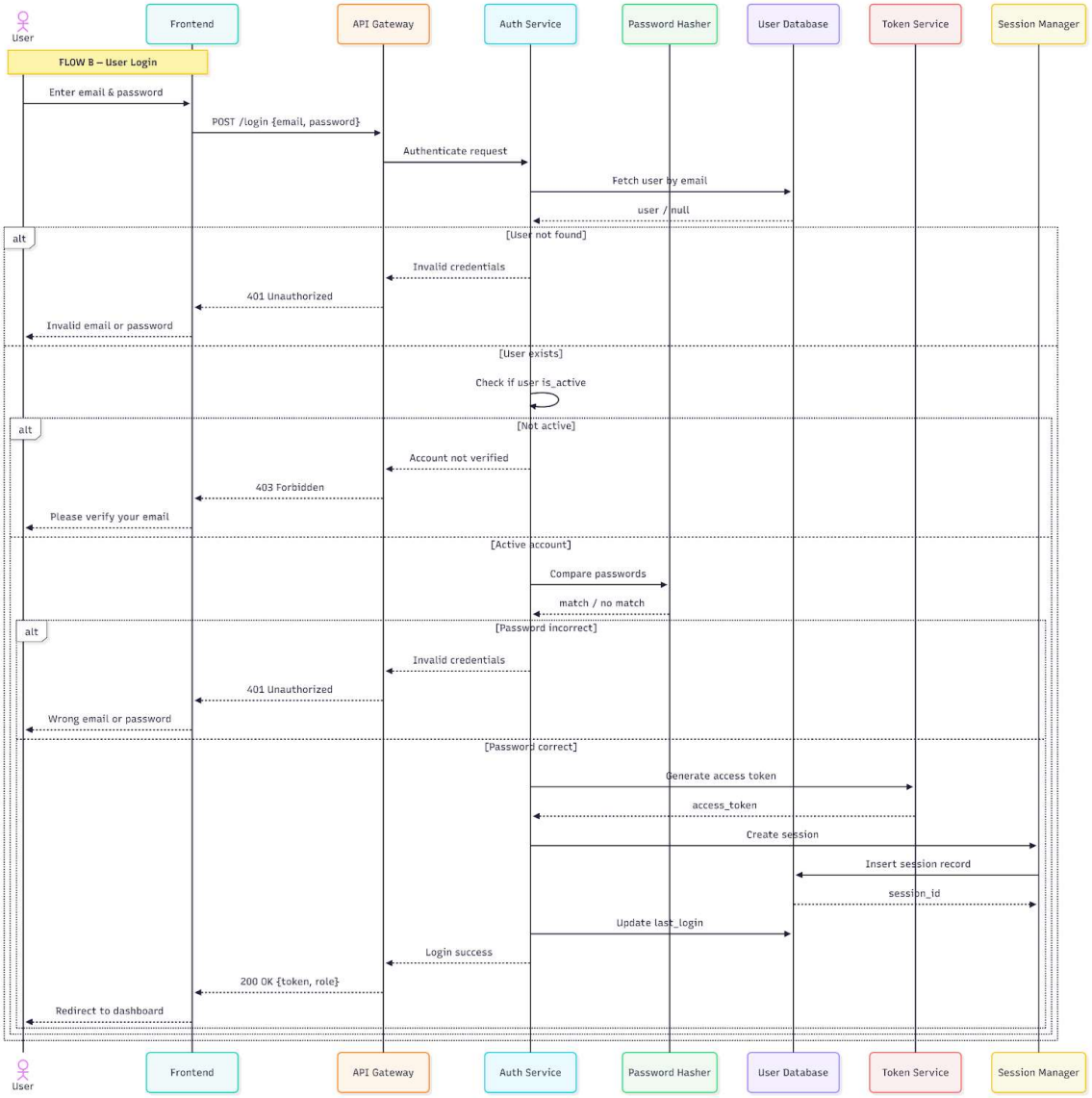


Fig 1.2B Flow B User Login

5.2.1.3 View Scan Results Dashboard

This sequence diagram illustrates how users interact with completed scan results, showing data retrieval, caching strategies, visualization generation, and interactive filtering capabilities. It demonstrates the user-facing output layer of the CloutCheck AI platform.

Phase 1: Loading Scan List

The interaction begins when a Brand Manager navigates to the "Scan Results" page. The Frontend Dashboard first validates the authentication token stored in localStorage to ensure the user is logged in. If valid, a GET request is sent to the API Gateway requesting all scans belonging to the authenticated user.

The API Gateway verifies the JWT token's validity and extracts the user_id. The request is forwarded to the Dashboard Service, which queries the MongoDB database for all ScanJob records where the user_id

matches. The query returns an array of scan summaries including scan_id, influencer profile name, scan date, and completion status.

This list is returned through the API to the frontend, where it's rendered as a sortable table. Users can see all their historical scans, sort by date or status, and select a specific scan for detailed analysis.

Phase 2: Viewing Detailed Results

When the user clicks on a specific scan, the frontend sends a GET request with the scan_id to retrieve detailed results. The API Gateway first verifies that the requesting user is authorized to view this scan (ownership check). If unauthorized, a 403 Forbidden response is returned.

For authorized requests, the Dashboard Service first checks the Cache Layer (Redis) using a key like "scan_{scan_id}". If the data exists in cache (cache hit), it's immediately returned, significantly reducing database load and response time for frequently accessed scans.

On a cache miss, the service executes multiple database queries to retrieve:

- Overall ReputationScore and category-specific scores
- All RiskAnalysis records showing detected risks by modality
- Flagged Content samples with evidence (text snippets, image URLs, timestamps)
- Platform distribution statistics (content count per platform)
- Timeline data showing risk patterns over time

This aggregated data is then stored in the cache for future requests.

The Dashboard Service passes the data to the Chart Generation Service, which prepares visualization configurations for:

- Risk score gauge (0-100 dial chart)
- Category breakdown pie chart (hate speech %, violence %, nudity %, etc.)
- Timeline graph showing risk trends over the content collection period
- Platform distribution bar chart showing content sources

All results, including chart configurations and data, are packaged and returned to the frontend as a comprehensive JSON response.

Phase 3: Rendering the Dashboard

The frontend receives the data and renders multiple dashboard components in parallel:

- A prominent risk score gauge at the top showing the overall reputation score
- Category breakdown charts visualizing which risk types were most prevalent
- A gallery of flagged content samples with thumbnails and severity indicators
- A timeline graph showing when risky content was posted
- Platform statistics showing which social media sources contained the most risks

The user can now comprehensively understand the influencer's reputation risk profile at a glance.

Phase 4: Interactive Filtering

Users can interact with the dashboard through filtering and sorting controls. When a user selects a filter (e.g., "Show only Hate Speech"), the frontend updates its local state and sends a filtered query to the API.

The Dashboard Service applies the filter condition to the database query, returning only Content and RiskAnalysis records matching the specified category. The frontend updates the content gallery to show only filtered results.

Similarly, when sorting by severity, the frontend can either sort client-side (for small datasets) or request server-side sorting for larger datasets. The API returns results ordered by severity_level in descending order.

Phase 5: Viewing Evidence Details

When a user clicks "View Evidence" on a specific flagged item, a modal opens. The frontend requests detailed evidence for that specific content item.

The Dashboard Service retrieves the RiskAnalysis record, which contains a JSON evidence field with:

- Exact text snippets that triggered the flag
- Bounding boxes showing where in images/videos the harmful content appears
- Timestamps indicating when in a video the issue occurs
- Confidence scores and model explanations

This detailed evidence is displayed in a modal, allowing users to verify the AI's findings and make informed decisions about the risk assessment.

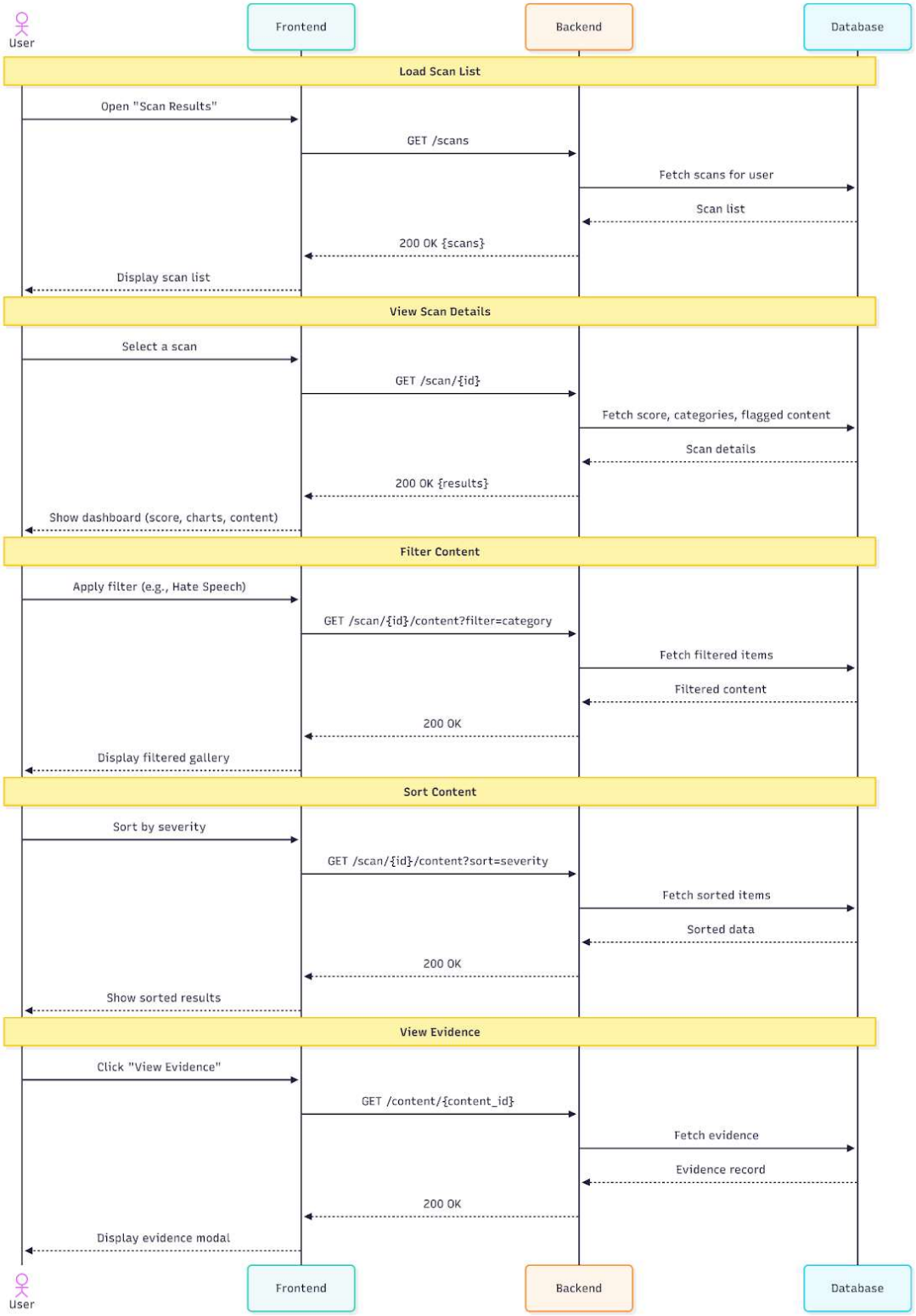


Fig 1.3 View Scan Results Dashboard

5.2.2 State Diagram

5.2.2.1 Scan Job Lifecycle State Diagram

This state diagram models the complete lifecycle of a scan job from creation to completion or failure. It represents the core business process of CloutCheck AI and shows how scans progress through various processing stages, including error handling and retry logic.

The scan job lifecycle begins when a user initiates a new scan through the dashboard. The ScanJob entity is created in the database with an initial state of Queued. In this state, the job is waiting in the processing queue for available worker resources. When the Content Scraper service picks up the job, it transitions to the Processing state. The ScanJob record is updated in the database, and active scraping begins. If the user cancels the scan before processing starts, the job transitions from Queued directly to Cancelled and then to the terminal state, with no further processing.

Within the Processing state, the scan enters a sub-state of ScrapingContent, where the system collects posts, images, videos, and audio from the specified social media platforms. If scraping encounters errors (API rate limits, network timeouts, invalid handles, or private profiles), the job transitions to the Failed state with appropriate error logging.

Upon successful content collection, the job transitions to AnalyzingText, where NLP models (BERT/RoBERTa) process textual content for hate speech, toxicity, and sentiment. If the NLP service experiences errors (model unavailable, timeout, or processing exceptions), the job transitions to Failed.

Following text analysis, the job moves to AnalyzingImages, where computer vision models (YOLOv8/ResNet) examine images and video frames for explicit content, violence, drugs, and weapons. Vision service errors also trigger a transition to Failed.

The next stage is AnalyzingAudio, where speech is transcribed using Whisper and analyzed for toxic language. Speech recognition failures or audio processing errors lead to the Failed state.

Once all modality-specific analyses succeed, the job transitions to FusingResults, where the Fusion Engine combines insights from text, image, and audio analyses using attention-guided fusion strategies. Fusion errors (such as incompatible data formats or mathematical exceptions) also result in failure.

The final processing stage is CalculatingScore, where the Risk Scoring Service computes weighted category scores and generates the overall reputation index (0-100). If scoring completes successfully, the job transitions to Completed, its final successful state. The results are now available for users to view on the dashboard.

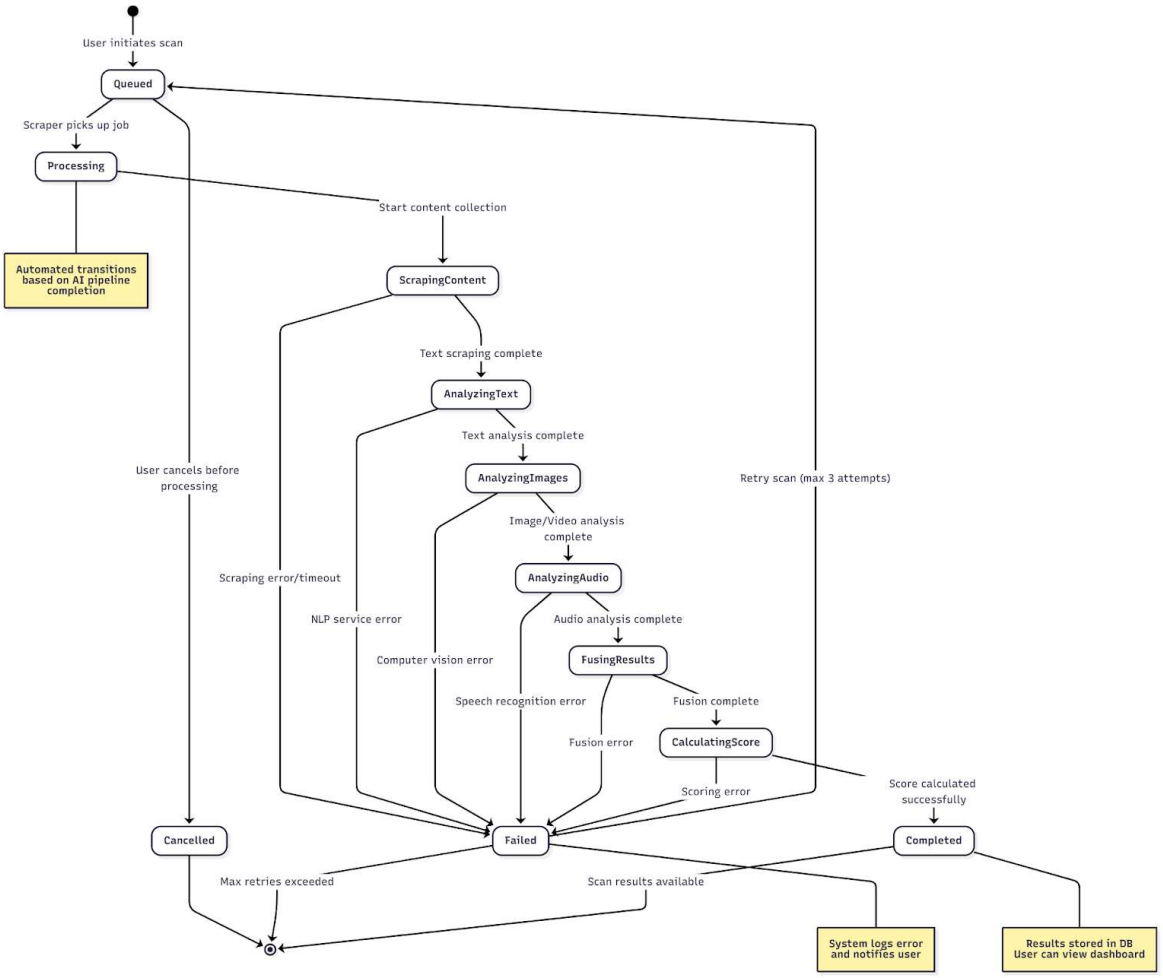


Fig 2.1 Scan Job Lifecycle

5.2.2.2 User Account Lifecycle State Diagram

This state diagram represents the lifecycle of a user account from registration through various active states to eventual deletion. It illustrates security policies, account verification requirements, suspension mechanisms, and dormancy handling.

The user account lifecycle begins when a new user completes the registration form. Upon submission, a User record is created in the database with an initial state of Unverified. In this state, the user cannot access any platform features. The account is essentially inactive until email verification is completed.

The system generates a verification token and sends it to the registered email address. This token has a 24-hour validity period. When the user clicks the verification link in the email, and the token is successfully validated, the account transitions to Active state. Users in the Active state have full access to all CloutCheck AI features including creating scans, viewing dashboards, generating reports, and managing profiles.

If the user does not verify their email within 24 hours, the verification token expires, and the account transitions to an Expired verification state. Users can request a new verification email, which transitions the account back to Unverified with a new token. If the account remains unverified for 7 days total, it is automatically deleted to prevent database pollution with abandoned registrations.

Suspended accounts transition to the Suspended state. Users in this state cannot log in, and all active sessions are terminated. A notification is sent explaining the suspension reason and any appeal process available.

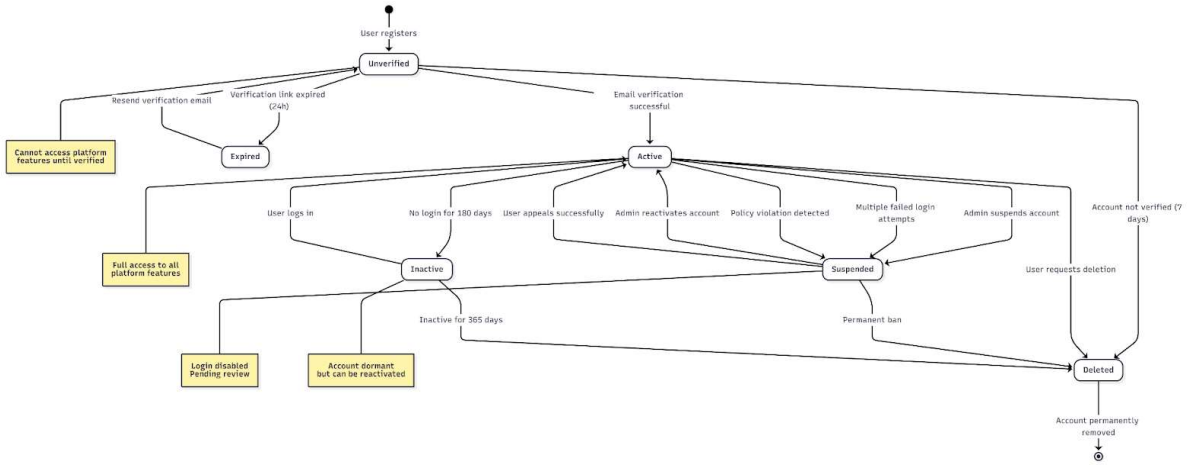


Fig 2.2 User Account Lifecycle

5.2.2.3 Report Generation Lifecycle State Diagram

This state diagram models the lifecycle of a reputation risk report from initial generation request through successful creation or failure. It demonstrates the asynchronous document generation process, error handling, and report availability management.

The report generation lifecycle begins after a scan job has completed successfully. Initially, no report exists, so the report entity is in the NotGenerated state. This is the default state for any completed scan that hasn't yet had a PDF report created.

When a user clicks the "Generate Report" button on the scan results dashboard, the report transitions to the Generating state. A report generation job is queued for asynchronous processing. The user sees a loading indicator with an estimated completion time of 10-30 seconds, depending on the amount of flagged content and complexity of visualizations.

When the user clicks "Download Report," the report transitions to Downloading state. The backend generates a signed URL (valid for 15 minutes) providing temporary access to the PDF file. The user's browser initiates the download.

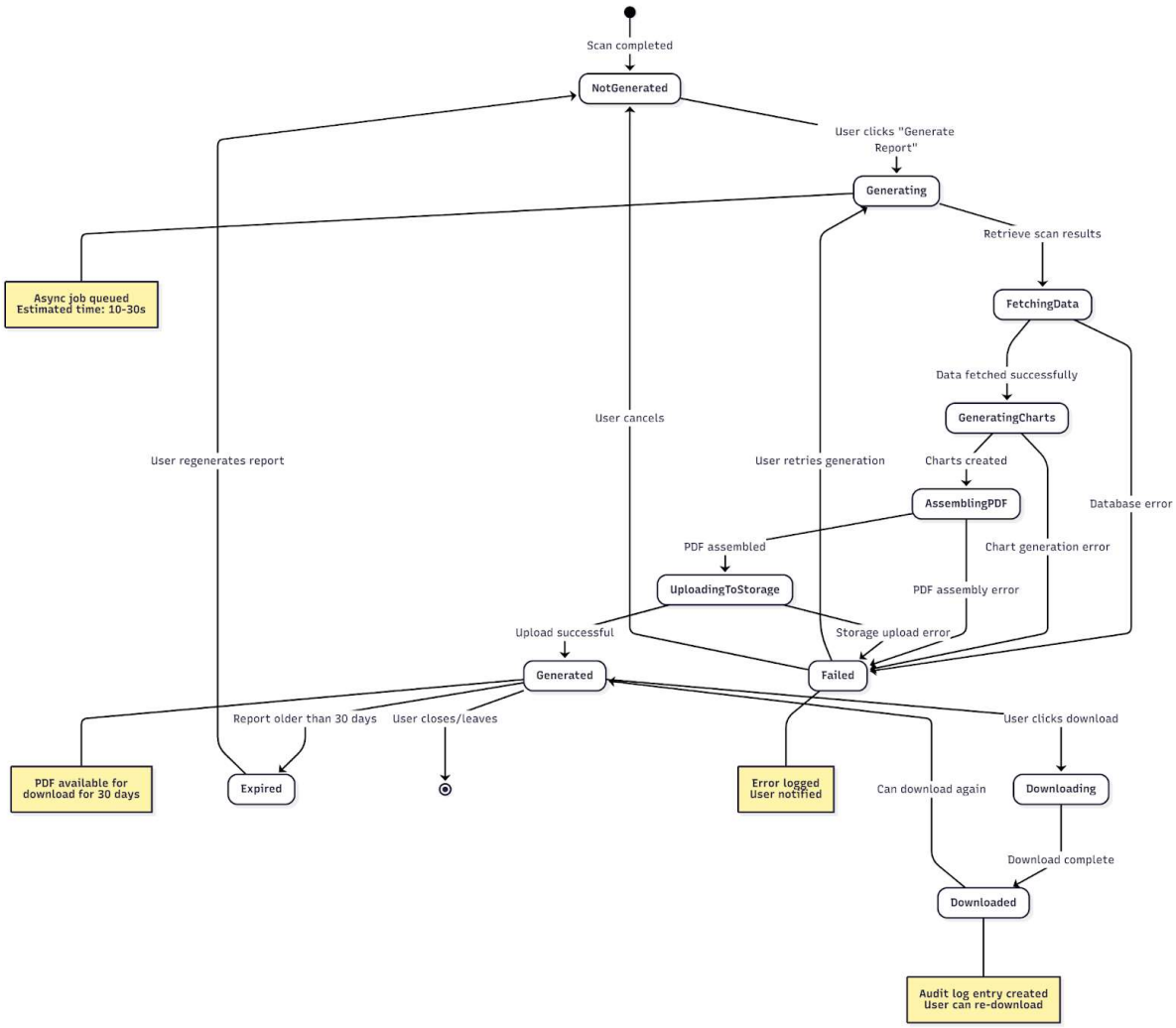


Fig 2.3 Report Generation LifeCycle

6 Appendices

Appendix A: API Endpoint Specifications

Authentication Endpoints

POST /api/auth/register

- *Request Body: { email, password, first_name, last_name, organization? }*
- *Response: { user_id, token, user: {...} }*
- *Status Codes: 201 Created, 400 Bad Request, 409 Conflict*

POST /api/auth/login

- *Request Body: { email, password }*
- *Response: { token, user: {...} }*

- *Status Codes: 200 OK, 401 Unauthorized*

POST /api/auth/logout

- *Headers: Authorization: Bearer {token}*
- *Response: { message: "Logged out successfully" }*
- *Status Codes: 200 OK*

Scan Endpoints

POST /api/scans

- *Headers: Authorization: Bearer {token}*
- *Request Body: { target_handle, platform, scan_type? }*
- *Response: { scan_id, status, initiated_at }*
- *Status Codes: 201 Created, 400 Bad Request, 401 Unauthorized*

GET /api/scans/{scan_id}

- *Headers: Authorization: Bearer {token}*
- *Response: { scan_id, status, overall_risk_score?, content_count, ... }*
- *Status Codes: 200 OK, 404 Not Found*

GET /api/scans

- *Headers: Authorization: Bearer {token}*
- *Query Params: ?page=1&limit=10&status=completed*
- *Response: { scans: [...], total, page, limit }*
- *Status Codes: 200 OK*

Risk Score Endpoints

GET /api/scans/{scan_id}/risks

- *Headers: Authorization: Bearer {token}*
- *Response: { risk_scores: [{ category, severity_score, confidence, ... }], ... }*
- *Status Codes: 200 OK, 404 Not Found*

GET /api/scans/{scan_id}/content

- *Headers: Authorization: Bearer {token}*
- *Query Params: ?page=1&limit=20&content_type=video*
- *Response: { content_items: [...], total, page, limit }*
- *Status Codes: 200 OK*

Report Endpoints

POST /api/reports

- *Headers: Authorization: Bearer {token}*
- *Request Body: { scan_id, report_type? }*
- *Response: { report_id, status, file_url? }*
- *Status Codes: 201 Created, 400 Bad Request*

GET /api/reports/{report_id}

- *Headers: Authorization: Bearer {token}*
- *Response: { report_id, status, file_url, generated_at, ... }*
- *Status Codes: 200 OK, 404 Not Found*

Appendix B: AI Model Specifications

Text Analysis Models

***Model**: RoBERTa-base fine-tuned on hate speech datasets*
***Model**: BERT-base for sentiment analysis*

Vision Analysis Models

***Model**: YOLOv8 for object detection*
***Model**: ResNet-50 fine-tuned for NSFW classification or Torch Transformer*

Audio Analysis Models

***Model**: OpenAI Whisper (base model)*

Appendix C: Deployment Architecture

***Cloud Provider**: AWS / Google Cloud Platform*

***Components**:*

- ***API Gateway**: AWS API Gateway or Google Cloud Endpoints*
- ***Compute**:*
  - *Frontend: Vercel or AWS Amplify*
  - *Backend: AWS ECS (Fargate) or Google Cloud Run*
  - *AI Workers: GPU-enabled EC2/Compute Engine instances*
- ***Database**: MongoDB Atlas*
- ***Storage**: AWS S3 or Google Cloud Storage*
- ***Queue**: AWS SQS + Celery or Google Cloud Tasks*
- ***Cache**: Redis (AWS ElastiCache or Google Memorystore)*
- ***Monitoring**: AWS CloudWatch or Google Cloud Monitoring*
- ***CI/CD**: GitHub Actions + Docker*

Appendix D: Security Measures

1. ***Authentication**: JWT tokens with 24-hour expiry*
2. ***Password Storage**: Bcrypt hashing (cost factor 12)*
3. ***API Rate Limiting**: 100 requests/minute per user*
4. ***Input Validation**: Pydantic models for all API inputs*
5. ***SQL/NoSQL Injection**: Parameterized queries, input sanitization*
6. ***Secrets Management**: AWS Secrets Manager or environment variables*
7. ***Data Encryption**: AES-256 for sensitive data at rest*
8. ***Audit Logging**: All user actions logged to AuditLogs collection*

Appendix E: Testing Strategy

***Unit Tests**:*

- *All backend services (80%+ code coverage)*
- *AI model inference functions*
- *Database operations*

***Integration Tests**:*

- *API endpoint workflows*
- *Database transactions*
- *External API integrations*

***End-to-End Tests**:*

- *User registration and login*
- *Complete scan workflow*
- *Report generation and download*

***Performance Tests**:*

- *Load testing (100 concurrent users)*
- *Stress testing (peak load scenarios)*
- *AI model inference latency*

***Tools**: pytest, Jest, Selenium, Locust*