Functional Specification — Corporate Videographer Search & Ranking System (v1.3)

Purpose Enable clients searching for corporate videographers to view ranked results based on relevance, reliability, and fairness — not just raw review averages. The system must surface the best-matched professionals for each brief using structured data, verified performance metrics, and contextual project cues.

1. Search Flow Overview When a client searches for “Corporate Videographer,” the system executes a multi-stage filtering and ranking pipeline before displaying results.

2. Filtering Workflow Stage 1 — Hard Constraints Filter out candidates failing non-negotiable project criteria. Include only videographers who: - List “Corporate Videos” in their service categories. - Are available on the requested project date/time. - Operate within the client’s location radius or offer remote service. - Have a price range overlapping the client’s stated budget. Outcome: Narrowed pool of qualified candidates.

Stage 2 — Relevance Analysis (Tag Matching) Measure alignment between client needs and videographer capabilities. Extract descriptive tags from the client brief (e.g. “boardroom,” “CEO interview,” “product launch”). Compare against portfolio tags (auto-extracted from past projects) and profile tags (declared skills, styles, niches). Compute a Tag Similarity Score (0–100%) using cosine or semantic similarity metrics. Outcome: Quantified relevance for each candidate.

Stage 3 — Reliability Weighting (Wilson Score) Adjust for statistical confidence in review data. Use Wilson score interval or equivalent metric. Reward consistent performance and higher review volume. Penalize outliers with limited data to reduce gaming. Outcome: Normalized reliability score across all sellers.

3. Final Ranking Computation Final_Score = (Tag_Similarity × 0.70) + (Reliability_Score × 0.30) Outcome: Unified, sortable list ordered by total fitness for the brief.

4. Client-Facing Output Each result displays: Videographer, Tag Match, Reliability, Price Range. Example: Corporate Films Co. — 95%, 4.9/5, £2k–£5k Studio Alpha — 85%, 4.8/5, £1.5k–£4k Urban Media — 90%, 4.2/5, £1k–£3k

UX Requirements - Show reasoning (“Best balance of relevance and reliability”). - Sorting: Best Match / Top Rated / Most Affordable. - Use visual indicators (progress bars or badges).

5. Integrity & Anti-Gaming - Review Validation: Weight by count to prevent inflation. - Tag Verification: Match against portfolio content. - Price Transparency: Enforce ≤ 2.5× spread between min/max. - Audit Logging: Track edits for anomaly detection.

6. Edge Case Handling New Videographers: Eligible if Tag Similarity ≥ 90%; show “Rising Talent” badge (<3 reviews). Ties: Break by total reviews → earliest join date → alphabetical.

7. Client Customization Modes - Best Match (default): Tag + Reliability (70/30) - Most Affordable: Price ascending - Top Rated: Reliability only

8. System Goals Ensure balanced, transparent ranking. Promote fair competition and trust. Allow flexible client control without revealing algorithm weights.

9. Implementation Guidance - Data model: category, availability, location, price_range, tags, reviews, reliability_score. - Tag extraction pipeline for briefs and portfolios. - Modular scoring logic; adjustable weights. - Clear UI showing match %, reliability, price. - Logging & monitoring for anomalies.

10. Success Metrics Top-3 matches accurate ≥ 90% Fraudulent review submissions < 5% Client satisfaction ≥ 4.5/5

11. Next Steps Confirm schema fields and API contracts. Define tag extraction + review scoring interfaces. Build prototype with test data. Run closed beta; gather QA metrics. Tune weights based on results.

12. Data Schema (Minimum Fields) id (UUID), name, categories, availability, location, price_range, portfolio_tags, profile_tags, reviews, reliability_score, join_date, status

13. Module Interfaces Tag Extraction Module: input (brief, portfolio), output (tags array) Reliability Module: input (reviews), output (reliability_score) Ranking Module: input (tag_similarity, reliability_score), output (final_score, ranked list)