Product Requirements Document (PRD): Bible Verse Auto-Detection SaaS Web App (MVP)

1. Overview

1.1 Product Name

VerseProjection (working title)

1.2 Purpose

The VerseProjection SaaS web app continuously analyzes a preacher's or presenter's speech during church services, mapping it to public-domain Bible texts (King James Version, World English Bible) to detect matching verses. When a spoken phrase aligns with a scripture (explicitly, e.g., "John 3:16," or paraphrased, e.g., "God so loved the world"), the app outputs the top 10 best-matching verses in the admin dashboard. Users select the correct verse, which is projected on a screen via a projector. The AI model learns from user selections to improve matching accuracy over time. An optional Electron-based desktop app supports offline use in low-internet environments. The MVP focuses exclusively on Bible verse detection and projection, ensuring simplicity and legal compliance.

1.3 Objectives

- Provide real-time, automated projection of Bible verses to enhance sermon delivery.
- Minimize manual effort for church tech teams (primary persona: small/medium church volunteers, ages 25–50).
- Achieve 90%+ detection accuracy and <2-second end-to-end latency (speech to projection).
- Output top 10 matching verses for user selection, supporting both explicit and paraphrased detections.
- Improve model accuracy via user feedback (selections stored for retraining).
- Ensure legal compliance using public-domain KJV/WEB texts.
- Scale for 1,000–10,000 simultaneous users (SOM: 10,000 organizations).

- Launch MVP in 6–9 months within a $150K–$400K budget.

1.4 Target Audience

- **Primary**: Church worship/tech teams (50–500 members, moderate technical proficiency, urban/suburban).
- **Secondary**: Megachurch media directors (500+ members, high technical expertise, urban).
- **Fringe**: Community/educational religious groups (low budgets, global).

1.5 Success Metrics

- **Adoption**: 1,000 organizations in Year 1, 10,000 in 3 years (SOM).
- **Performance**: 90%+ detection accuracy, <2s latency (speech to projection), top 10 matches include correct verse in 95%+ cases.
- **Model Improvement**: 5% accuracy increase after 3 months of user feedback.
- **Revenue**: $2M/year from 10,000 organizations at $200/year subscription.
- **User Satisfaction**: 80%+ positive feedback from pilot churches (NPS > 50).
- **Reliability**: 99.9% uptime during peak usage (Sunday services).

2. Market and Competitive Context

2.1 Market Opportunity

- **TAM**: 1M religious organizations globally using projection software ($200M at $200/year).
- **SAM**: 200,000 digitally equipped churches ($40M).
- **SOM**: 10,000 organizations in 3 years ($2M).
- **Source**: The Lead Pastor (2023), Barna Group (2020), Capterra (2025).

2.2 Competitive Landscape

- **Competitors**: ProPresenter, EasyWorship, MediaShout, OpenLP, Quelea.
- **Differentiator**: Real-time speech analysis for automated Bible verse detection and projection, with top 10 match selection and continuous learning, unlike manual slide-based competitors.
- **Gap**: Competitors lack automation for spontaneous verse references, requiring pre-prepared slides.

2.3 Licensing Considerations

- **Bible Texts**: Public-domain KJV and WEB are freely downloadable and storable from repositories like CrossWire Bible Society, Project Gutenberg, or Bible Gateway (JSON exports). No licensing costs or legal risks.
- **Proprietary Bibles**: Deferred to premium tiers (e.g., NIV, ESV via Zondervan, $1,000–$10,000/year).
- **Song Lyrics**: Excluded from MVP to avoid CCLI licensing or copyright issues with external APIs or AI models.

3. Functional Requirements

3.1 Core Features

1. **Real-Time Audio Capture**:
    - Capture continuous audio from a USB microphone or church soundboard via Web Audio API.
    - Support 16-bit, 44.1 kHz audio for high-quality speech detection.
    - Apply client-side noise filtering (e.g., via RecordRTC) to handle church environments (clapping, ambient noise, music).
2. **Verse Detection**:
    - Continuously transcribe preacher's speech using OpenAI Whisper (90%+ accuracy in noisy settings).
    - Map transcribed text to KJV/WEB verses using BERT, identifying matches for explicit references (e.g., "John 3:16") or paraphrased quotes (e.g., "God so loved the world").
    - Output up to 10 best-matching verses ranked by semantic similarity (cosine similarity score, 95%+ chance correct verse in top 10).
    - Process in <1 second (transcription: ~500ms, matching: ~500ms).
    - Support two detection scenarios:
        - **Explicit References**: Preacher cites a verse (e.g., "Please turn to John 3:16"), easily detected via regex and exact matching.
        - **Paraphrased Quotes**: Preacher quotes scripture indirectly (e.g., "God loved the world and gave His Son"), detected via semantic similarity.
3. **Verse Selection and Projection**:
    - Display top 10 matching verses in the admin dashboard with verse text, reference, and confidence score (e.g., "John 3:16 – KJV – 92%").
    - Allow user to click the correct verse for immediate projection on a secondary browser window (full-screen mode) via HDMI/VGA.
    - Support customizable display (font size, color, background).

    - Achieve <2-second end-to-end latency (speech to projection, including user selection).
4. **Model Learning**:
    - Store user-selected verses in PostgreSQL as feedback data (e.g., transcription, selected verse, confidence scores).
    - Use feedback to incrementally retrain BERT, improving matching accuracy over time (target: 5% accuracy increase after 3 months).
    - Implement periodic retraining (e.g., monthly) via AWS SageMaker.
5. **Admin Dashboard**:
    - Configure settings (Bible version: KJV/WEB, display options, audio input).
    - Display real-time transcription and top 10 verse matches with clickable selection.
    - Provide manual override via search bar (e.g., enter "John 3:16") for misdetections.
    - Show detection history and confidence metrics for monitoring.
6. **Offline Mode (Desktop App)**:
    - Cache KJV/WEB texts (~4MB) and lightweight Whisper/BERT models in Electron app.
    - Perform local audio capture, transcription, and matching (CPU-based).
    - Display top 10 matches for user selection, project verses locally.
    - Store user selections locally, syncing feedback to cloud when online for model retraining.

3.2 Non-Functional Requirements

- **Performance**: <2s end-to-end latency, 90%+ detection accuracy, top 10 matches include correct verse in 95%+ cases.
- **Scalability**: Support 1,000–10,000 simultaneous users during peak times (Sunday services).
- **Reliability**: 99.9% uptime via AWS auto-scaling.
- **Security**: HTTPS for communication, JWT for authentication, AES-256 encryption for database and feedback data.
- **Compatibility**: Chrome/Firefox browsers, 720p/1080p projectors, Windows/macOS laptops (8GB RAM, Core i5).
- **Bandwidth**: <500 kbps for audio streaming (Opus codec).
- **Learning**: Model improves by 5% in accuracy after 3 months of user feedback.

4. Technical Architecture

4.1 Architecture Overview

- **Frontend**: React web app for projection, dashboard, and verse selection; Electron for offline desktop app.
- **Backend**: Node.js/Express for APIs, PostgreSQL for storing KJV/WEB texts and user feedback.
- **AI/ML Pipeline**: OpenAI Whisper for continuous speech-to-text, BERT for semantic verse matching, hosted on AWS SageMaker, optimized with ONNX Runtime.
- **Cloud**: AWS (EC2, RDS, SageMaker, S3, API Gateway, CloudWatch) for hosting, scaling, and logging.
- **Data Flow**: Audio → WebSocket → Backend → AI (transcription, top 10 matches) → Frontend (user selection) → Database (feedback storage) → Frontend (projection).

4.2 Connection Flow

1. **Login**:
    - User navigates to `app.verseprojection.com` in Chrome/Firefox on a laptop connected to a projector.
    - Logs in via email/password or SSO (AWS Cognito, JWT authentication).
    - Accesses dashboard to configure settings (e.g., KJV/WEB, font size, audio input).
2. **Service Activation**:
    - User clicks "Start Projection" to initialize WebSocket (Socket.IO) and AI pipeline.
3. **Audio Capture**:
    - Web Audio API captures continuous audio from a USB microphone or soundboard (16-bit, 44.1 kHz).
    - Client-side JavaScript (RecordRTC) applies noise filtering (e.g., high-pass filter at 100 Hz) to reduce background noise (clapping, music).
    - User selects input device in dashboard (dropdown, e.g., "USB Mic", "Soundboard Line-In").
4. **Audio Streaming**:
    - Audio is compressed (Opus codec, <500 kbps, 20ms frames) and streamed via WebSocket to the Node.js backend.
    - AWS API Gateway and Elastic Load Balancer manage concurrent streams from multiple churches.
5. **Audio Analysis**:

- **Transcription**: OpenAI Whisper transcribes audio in real-time (~500ms per 5-second chunk) on AWS SageMaker, fine-tuned for sermon speech (e.g., verse references, religious vocabulary, accents).
- **Verse Matching**:
  - BERT processes transcribed text (~500ms), generating embeddings (e.g., 768-dimensional vectors) for semantic similarity against KJV/WEB verses.
  - For explicit references (e.g., "John 3:16"), apply regex (e.g., `[A-Za-z]+ \d+:\d+`) for exact matching, bypassing full semantic analysis (<100ms).
  - For paraphrases, compute cosine similarity between transcription and verse embeddings, ranking top 10 matches (95%+ chance correct verse in list).
  - Output includes verse reference, text, version (KJV/WEB), and confidence score (e.g., 92%).
- Total processing: <1 second (transcription: ~500ms, matching: ~100–500ms).

6. **Verse Selection**:
   - Top 10 matches are sent via WebSocket to the React frontend and displayed in the admin dashboard (e.g., table with columns: Reference, Text, Version, Confidence).
   - User clicks the correct verse (e.g., "John 3:16 – KJV"), triggering immediate projection.
   - Selection is logged in PostgreSQL (transcription, selected verse, confidence scores) for model retraining.
7. **Projection**:
   - Selected verse is rendered on a secondary browser window (full-screen, 720p/1080p) connected to a projector via HDMI/VGA.
   - Total latency (speech to projection, including user selection): <2 seconds.
8. **Monitoring and Overrides**:
   - Dashboard shows real-time transcription, top 10 matches, and selection history.
   - Manual override via search bar (e.g., type "John 3:16") for misdetections.
   - AWS CloudWatch logs errors (e.g., low-confidence matches, transcription failures) for debugging.
9. **Model Learning**:
   - User selections are stored in PostgreSQL (`feedback` table).

- Monthly retraining job on AWS SageMaker updates BERT weights using feedback data (e.g., transcription → selected verse pairs), improving similarity scoring.

4.3 Key Modules

1. **Audio Input Module**:
    - **Technologies**: Web Audio API, PyAudio (desktop app), RecordRTC, OpenAI Whisper.
    - **Function**: Captures and preprocesses continuous audio from microphone or soundboard.
    - **Requirements**:
        - 16-bit/44.1 kHz quality for clear speech.
        - Noise filtering (e.g., 100 Hz high-pass, 8 kHz low-pass) for church environments.
        - <200ms capture latency, <500 kbps streaming bandwidth.
2. **NLP/Text Matching Module**:
    - **Technologies**: BERT (Hugging Face Transformers), PostgreSQL, ONNX Runtime, regex for explicit references.
    - **Function**:
        - Transcribes speech via Whisper (90%+ accuracy).
        - Matches text to KJV/WEB verses, outputting top 10 matches (95%+ correct verse inclusion).
        - Handles explicit references (regex, <100ms) and paraphrases (semantic similarity, ~500ms).
    - **Requirements**:
        - <1-second processing (transcription: ~500ms, matching: ~100–500ms).
        - Fine-tuned for sermon speech and verse paraphrases.
        - Confidence scores for ranking (e.g., cosine similarity 0–1).
3. **Database Module**:
    - **Technologies**: PostgreSQL with full-text search and vector extensions (e.g., pgvector for embeddings).
    - **Function**: Stores KJV/WEB verses, user settings, and feedback data for retraining.
    - **Requirements**:
        - ~4MB for KJV/WEB texts, sub-100ms query latency.
        - Supports vector similarity searches for BERT embeddings.
        - Sourced from public-domain repositories (e.g., CrossWire Bible Society).

        - AES-256 encryption for data at rest.
4. **Projection Display Module**:
    - **Technologies**: React, Socket.IO, Electron (offline).
    - **Function**: Renders user-selected verse on projector (720p/1080p) via full-screen browser window.
    - **Requirements**:
        - Customizable (font size: 12–48pt, color, background).
        - <200ms rendering latency.
        - Compatible with HDMI/VGA projectors.
5. **Admin Dashboard Module**:
    - **Technologies**: React, Material-UI.
    - **Function**:
        - Configures settings (Bible version, display, audio input).
        - Displays real-time transcription and top 10 verse matches (clickable table).
        - Provides manual override and detection history.
    - **Requirements**:
        - Intuitive for non-technical users (drag-and-drop settings, clear buttons).
        - Real-time updates via WebSocket.
6. **Cloud Integration Module**:
    - **Technologies**: AWS (EC2, RDS, SageMaker, S3, API Gateway, CloudWatch).
    - **Function**: Hosts backend, AI, database; scales for peak usage; manages retraining.
    - **Requirements**:
        - 99.9% uptime.
        - $400–$800/month cost for 1,000 users.
        - Auto-scaling for 1,000–10,000 users.

5. Skillset Requirements

5.1 Roles and Responsibilities

1. **AI/ML Engineer**:
    - **Skills**: Python, TensorFlow/PyTorch, OpenAI Whisper, BERT, ONNX, audio processing (Librosa), incremental learning.
    - **Tasks**:
        - Develop/fine-tune Whisper for sermon transcription.

        - Fine-tune BERT for verse matching (explicit and paraphrased).
        - Implement top 10 match ranking and feedback-based retraining.
        - Optimize pipeline for <1s latency using ONNX.
    - **Need**: Critical for detection accuracy and model learning.
2. **Backend Developer**:
    - **Skills**: Node.js/Express, PostgreSQL (pgvector), AWS (EC2, RDS, API Gateway).
    - **Tasks**:
        - Build APIs for audio streaming, verse matching, and feedback storage.
        - Manage database (KJV/WEB, feedback data).
        - Ensure scalability and security (JWT, AES-256).
    - **Need**: Essential for system reliability and data handling.
3. **Frontend Developer**:
    - **Skills**: React, Socket.IO, Material-UI/Tailwind CSS, Electron, Web Audio API.
    - **Tasks**:
        - Develop dashboard with top 10 match selection and override functionality.
        - Build projection UI and offline Electron app.
        - Ensure responsive, user-friendly design.
    - **Need**: Critical for user experience and adoption.
4. **DevOps Engineer (Part-Time)**:
    - **Skills**: AWS (EC2, RDS, SageMaker, CloudWatch), CI/CD (GitHub Actions).
    - **Tasks**:
        - Deploy app to AWS, optimize costs ($400–$800/month).
        - Set up auto-scaling and monitoring.
        - Manage SageMaker retraining jobs.
    - **Need**: Necessary for scalability and operational efficiency.
5. **Product Manager**:
    - **Skills**: Church technology knowledge, MVP scoping, user feedback analysis.
    - **Tasks**:
        - Define features (multi-match, learning).
        - Source public-domain Bible texts (e.g., CrossWire).
        - Coordinate pilot testing with churches.
    - **Need**: Essential for market alignment and user-centric design.

5.2 Team Size

- **Core Team**: 3–4 members (AI/ML Engineer, Full-Stack Developer, Part-Time DevOps, Product Manager).
- **Timeline**: 6–9 months (slightly increased complexity for multi-match and learning).
- **Budget**: $150K–$400K (salaries, AWS, tools).

6. Implementation Details

6.1 Technology Stack

- **Frontend**:
  - React (UI framework for dashboard, projection, and match selection).
  - Socket.IO (real-time WebSocket for audio, matches, and projection).
  - Material-UI/Tailwind CSS (responsive, user-friendly styling).
  - Electron (offline desktop app).
- **Backend**:
  - Node.js/Express (API server for audio, matching, feedback).
  - PostgreSQL (database for KJV/WEB, feedback, users).
- **AI/ML**:
  - OpenAI Whisper (continuous speech-to-text, fine-tuned for sermons).
  - BERT (Hugging Face Transformers, semantic matching, top 10 ranking).
  - TensorFlow/PyTorch (model training frameworks).
  - Librosa (audio preprocessing, e.g., noise reduction).
  - ONNX Runtime (low-latency inference).
- **Cloud**:
  - AWS EC2 (backend hosting, t3.medium, auto-scaling).
  - AWS RDS (PostgreSQL, db.t3.micro, 5GB storage).
  - AWS SageMaker (AI inference and retraining, ml.t3.medium).
  - AWS S3 (logs, feedback data backups).
  - AWS API Gateway (WebSocket management).
  - AWS CloudWatch (error logging, performance monitoring).
- **Tools**:
  - GitHub (version control).
  - GitHub Actions (CI/CD pipeline).
  - RecordRTC (client-side audio preprocessing).
  - Web Audio API (browser-based audio capture).

6.2 Database Schema

- **Bible Table**:

    - **Columns**:
        - `id` (UUID, primary key).
        - `version` (string, e.g., “KJV”, “WEB”).
        - `book` (string, e.g., “John”).
        - `chapter` (integer, e.g., 3).
        - `verse` (integer, e.g., 16).
        - `text` (text, e.g., “For God so loved the world…”).
        - `embedding` (vector, 768-dimensional BERT embedding for similarity).
    - **Size**: ~4MB for KJV/WEB texts, ~8MB with embeddings.
    - **Indexing**:
        - Full-text search index on `text` for manual overrides.
        - Vector index (pgvector) on `embedding` for semantic similarity (cosine distance).
    - **Source**: Public-domain JSON exports from CrossWire Bible Society or Project Gutenberg.
    - **Security**: AES-256 encryption.
- **Users Table**:
    - **Columns**:
        - `id` (UUID, primary key).
        - `email` (string, unique).
        - `password_hash` (string, bcrypt).
        - `subscription_tier` (string, e.g., “free”, “premium”).
        - `settings` (JSON, e.g., {“bible_version”: “KJV”, “font_size”: 24}).
    - **Security**: AES-256 encryption, JWT for sessions.
- **Feedback Table**:
    - **Columns**:
        - `id` (UUID, primary key).
        - `user_id` (UUID, foreign key to Users).
        - `timestamp` (datetime).
        - `transcription` (text, e.g., “God loved the world”).
        - `selected_verse_id` (UUID, foreign key to Bible).
        - `top_matches` (JSON, e.g., [{“verse_id”: “uuid”, “confidence”: 0.92}, …]).
    - **Size**: ~10MB/month for 1,000 users (assuming 100 selections/user/month).
    - **Use**: Stores data for BERT retraining, improving accuracy.

        - **Security**: AES-256 encryption.
- **Logs Table** (optional, for debugging):
    - **Columns**:
        - `id` (UUID, primary key).
        - `timestamp` (datetime).
        - `user_id` (UUID, foreign key).
        - `event` (string, e.g., "low_confidence_match").
        - `details` (JSON, e.g., {"transcription": "text", "confidence": 0.6}).
    - **Size**: ~1MB/month for 1,000 users.
    - **Storage**: AWS S3 or CloudWatch.

6.3 AI Pipeline

- **Input**: Continuous audio stream (16-bit, 44.1 kHz, Opus codec, 5-second chunks).
- **Processing**:
    - **Whisper**:
        - Transcribes audio to text (~500ms per chunk).
        - Fine-tuned on sermon audio datasets (~10 hours, public-domain or synthetic) for religious vocabulary, verse references, and noise resilience (clapping, music).
        - Parameters: Whisper-small (~240M parameters, ~500MB memory).
    - **BERT**:
        - Processes transcribed text (~500ms), generating 768-dimensional embeddings.
        - For explicit references (e.g., "John 3:16"):
            - Regex pattern `[A-Za-z]+ \d+:\d+` extracts verse references.
            - Exact match against Bible table (<100ms).
        - For paraphrases:
            - Compute cosine similarity between transcription embedding and precomputed verse embeddings (stored in `Bible.embedding`).
            - Rank top 10 matches by similarity score (0–1, e.g., 0.92 for "John 3:16").
            - Cache embeddings in memory (Redis, ~8MB) for sub-100ms queries.

        - Parameters: BERT-base-uncased (~110M parameters, ~400MB memory).
    - **Optimization**:
        - ONNX Runtime reduces inference time (e.g., 30% faster on CPU).
        - Batch processing for overlapping audio chunks (e.g., 5s chunks every 2s).
    - **Output**:
        - Array of top 10 matches: [{`verse_id`, `reference`, `text`, `version`, `confidence`}, ...].
        - Sent to frontend via WebSocket.
- **Feedback Loop**:
    - User selections stored in `Feedback` table (transcription, selected verse, top matches).
    - Monthly SageMaker job retrains BERT using feedback data (~1,000–10,000 pairs/month for 1,000 users).
    - Loss function: Triplet loss to optimize embedding distances (correct verse closer, incorrect farther).
    - Target: 5% accuracy increase after 3 months.
- **Training Data**:
    - Whisper: ~10 hours of sermon audio (public-domain sermons, synthetic clips with verse references).
    - BERT: ~10,000 transcription-verse pairs (KJV/WEB texts, paraphrased examples, e.g., "God so loved" → John 3:16).
    - Feedback: Incremental user selections (e.g., 1,000 pairs/week for 1,000 users).
- **Hosting**: AWS SageMaker (ml.t3.medium instances, auto-scaling).

6.4 Cloud Infrastructure

- **AWS EC2**:
    - Hosts Node.js/Express backend.
    - Instance: t3.medium (2 vCPUs, 4GB RAM).
    - Auto-scaling: 1–10 instances for Sunday peaks.
    - Cost: ~$50–$200/month for 1,000 users.
- **AWS RDS**:
    - PostgreSQL database (KJV/WEB, feedback, users).
    - Instance: db.t3.micro (1 vCPU, 1GB RAM, 10GB storage for ~12MB data).
    - Extensions: pgvector for vector similarity searches.
    - Cost: ~$50–$100/month.
- **AWS SageMaker**:

    - Hosts Whisper and BERT for inference and retraining.
    - Instance: ml.t3.medium (2 vCPUs, 4GB RAM) for inference.
    - Retraining: ml.m5.large (2 vCPUs, 8GB RAM, ~1 hour/month).
    - Cost: ~$250–$450/month (inference: $200–$400, retraining: $50).
- **AWS S3**:
    - Stores logs, feedback backups, and optional audio recordings.
    - Cost: ~$10–$50/month.
- **AWS API Gateway**:
    - Manages WebSocket connections (audio, matches, selections).
    - Cost: ~$20–$50/month.
- **AWS CloudWatch**:
    - Logs errors (e.g., low-confidence matches, transcription failures).
    - Monitors latency, accuracy, and user selection rates.
    - Cost: ~$10–$20/month.
- **AWS ElastiCache (Redis)**:
    - Caches verse embeddings (~8MB) for sub-100ms similarity queries.
    - Instance: cache.t3.micro (1 vCPU, 0.5GB RAM).
    - Cost: ~$20–$30/month.
- **Total Cost**: $460–$900/month for 1,000 users, optimized with Reserved Instances.

6.5 Offline Mode (Desktop App)

- **Technologies**: Electron, lightweight Whisper/BERT models, SQLite, PyAudio.
- **Storage**:
    - ~4MB for KJV/WEB texts, ~8MB for precomputed embeddings (SQLite).
    - ~500MB for Whisper-small and BERT-base models (ONNX format).
- **Processing**:
    - Audio capture via PyAudio (16-bit, 44.1 kHz).
    - Local Whisper/BERT inference (CPU-based, 8GB RAM, Core i5).
    - Top 10 match ranking using SQLite vector searches (pgvector equivalent).
- **Functionality**:
    - Displays top 10 matches in Electron UI for user selection.
    - Projects selected verse in full-screen window.
    - Stores selections locally (SQLite `feedback` table).
    - Syncs feedback to cloud (PostgreSQL) when online for retraining.
- **Requirements**:
    - ~600MB disk space.
    - <2-second latency (local processing).

    - No internet dependency.

7. User Experience

7.1 User Flow

1. **Setup**:
    - Connect laptop to projector (HDMI/VGA).
    - Log into web app (`app.verseprojection.com`) or launch Electron app (offline).
2. **Configuration**:
    - Select KJV or WEB, audio input (mic/soundboard), display settings (font size: 12–48pt, color, background).
    - Adjust sensitivity (e.g., confidence threshold: 0.7–0.9) for match detection.
3. **Operation**:
    - Click "Start Projection"; app continuously analyzes preacher's speech.
    - Top 10 matching verses appear in dashboard (e.g., "John 3:16 – KJV – 92%").
    - User clicks correct verse for projection (e.g., ~1-second decision).
    - Verse displays on projector in full-screen window.
4. **Overrides**:
    - If no correct match, use search bar to manually select verse (e.g., "John 3:16").
    - Dashboard shows transcription and detection history for context.
5. **Offline**:
    - Electron app runs locally, mirroring web app functionality.
    - Syncs selections to cloud when reconnected.

7.2 UI Components

- **Login Page**:
    - Email/password or SSO (AWS Cognito).
    - Minimal design, dark/light theme toggle.
- **Dashboard**:
    - **Settings Panel**:
        - Dropdowns: Bible version (KJV/WEB), audio input, projector resolution (720p/1080p).
        - Sliders: Font size (12–48pt), confidence threshold (0.7–0.9).
        - Color picker: Text, background.

    - **Transcription View**:
        - Real-time text (e.g., "Pastor: 'God so loved the world'").
        - Scrollable, 10-second history.
    - **Matches Table**:
        - Columns: Reference (e.g., "John 3:16"), Text (preview, e.g., "For God so loved…"), Version (KJV/WEB), Confidence (e.g., 92%).
        - Clickable rows to select verse for projection.
        - Sorted by confidence (descending).
    - **Override Search Bar**:
        - Autocomplete for verse references (e.g., type "Jo" → "John").
        - Instant display on selection.
    - **Detection History**:
        - Timeline of past matches and selections (e.g., "10:02 AM: John 3:16 selected").
- **Projection Window**:
    - Full-screen browser or Electron window.
    - Customizable verse text (e.g., white text, black background, 24pt Arial).
    - Smooth transitions (200ms fade-in).
- **Error Notifications**:
    - Pop-ups for issues (e.g., "Low audio quality, check mic").
    - Suggests manual override for low-confidence matches (<0.7).

7.3 UX Principles

- **Simplicity**: One-click projection start, intuitive match selection (clickable table).
- **Reliability**: Real-time feedback (transcription, matches), robust overrides.
- **Accessibility**:
    - High-contrast text (WCAG 2.1 compliant).
    - Keyboard navigation (e.g., arrow keys for match selection).
    - Support for 720p/1080p projectors.

8. Constraints and Risks

8.1 Technical Constraints

- **Latency**: Must achieve <2s end-to-end latency (speech to projection, including user selection).
- **Noise**: Church environments (clapping, music, congregation) may reduce Whisper's transcription accuracy.

- **Multi-Match Accuracy**: Top 10 matches must include correct verse in 95%+ cases, requiring robust BERT fine-tuning.
- **Browser Compatibility**: Web Audio API and WebSocket vary across browsers (e.g., Safari limitations).
- **Hardware**: Laptops need 8GB RAM, Core i5; projectors require 720p/1080p support.
- **Learning**: Feedback loop must improve accuracy (5% in 3 months) without degrading performance.

8.2 Licensing Risks

- **Bibles**: Public-domain KJV/WEB texts are legally free to download and store (e.g., CrossWire Bible Society). No risks.
- **Proprietary Bibles**: Deferred to premium tiers to avoid licensing costs ($1,000–$10,000/year).

8.3 Mitigation Strategies

- **Noise**:
  - Fine-tune Whisper on sermon audio with noise (clapping, music, ~10 hours).
  - Apply advanced filtering (Librosa, high-pass at 100 Hz, low-pass at 8 kHz).
  - Provide manual override for transcription errors.
- **Multi-Match Accuracy**:
  - Fine-tune BERT on ~10,000 transcription-verse pairs, emphasizing paraphrases.
  - Use feedback data to retrain monthly, prioritizing high-confidence mismatches.
  - Set default confidence threshold (0.7) to balance sensitivity and precision.
- **Browser**:
  - Optimize for Chrome (80% church usage, Capterra 2025), test Firefox.
  - Provide setup guides for audio/WebSocket troubleshooting.
  - Fallback to Electron app for incompatible browsers.
- **Hardware**:
  - Support multiple projector resolutions (720p, 1080p).
  - Recommend HDMI/VGA adapters in user documentation.
  - Test Electron app on low-end laptops (8GB RAM, Core i5).
- **Learning**:

    - Validate feedback data (e.g., filter low-confidence selections).
    - Monitor retraining impact via CloudWatch (accuracy, latency).
    - Cap retraining to ~1 hour/month to control costs.
- **Licensing**:
    - Source KJV/WEB from verified repositories (CrossWire Bible Society, Project Gutenberg).
    - Document sourcing in app (e.g., “KJV sourced from CrossWire, public domain”).
- **Tarpit Risk**:
    - Focused scope (verses only) and public-domain texts minimize complexity.
    - Continuous analysis and multi-match selection add moderate complexity, mitigated by leveraging existing AI pipeline and robust testing.

9. Development Timeline

- **Month 1–2**:
    - Gather requirements, source KJV/WEB texts (CrossWire Bible Society).
    - Set up PostgreSQL database (Bible, Feedback tables).
    - Collect sermon audio for Whisper fine-tuning (~10 hours).
- **Month 3–4**:
    - Develop AI pipeline:
        - Fine-tune Whisper for sermon speech (noise, accents).
        - Fine-tune BERT for verse matching (explicit references, paraphrases).
        - Implement top 10 match ranking and feedback storage.
    - Build backend APIs (Node.js/Express) for audio, matching, feedback.
- **Month 5–6**:
    - Develop frontend (React):
        - Dashboard with match selection table, override search.
        - Projection window with customizable display.
    - Integrate WebSocket (Socket.IO) for real-time updates.
- **Month 7–8**:
    - Deploy to AWS (EC2, RDS, SageMaker, API Gateway, CloudWatch, ElastiCache).
    - Develop offline Electron app (SQLite, local Whisper/BERT).
    - Test in simulated church environments (noise, paraphrases, explicit references).

- **Month 8–9**:
  - Pilot with 5–10 churches (urban/suburban, 50–500 members).
  - Collect feedback data (~1,000 selections/pilot).
  - Iterate UI (e.g., simplify match selection) and model (first retraining).
  - Finalize MVP for launch.

10. Budget Allocation

- **Salaries**: $120K–$300K (3–4 team members, 6–9 months).
  - AI/ML Engineer: $50K–$100K.
  - Full-Stack Developer: $50K–$100K.
  - Part-Time DevOps: $10K–$30K.
  - Product Manager: $10K–$30K.
- **AWS Hosting**: $6K–$18K (EC2, RDS, SageMaker, S3, API Gateway, CloudWatch, ElastiCache).
  - Monthly: $460–$900 for 1,000 users.
  - Annual: ~$5.5K–$10.8K.
- **Tools/Software**: $5K (GitHub, Hugging Face, testing tools, audio datasets).
- **Testing/Pilot**: $2K–$5K (church outreach, feedback collection).
- **Total**: $153K–$428K (slightly increased due to feedback loop and multi-match complexity, within $150K–$400K target via cost optimization).

11. Sources

- Capterra (2025): Church software trends.
- Barna Group (2020): Church technology preferences.
- AWS Documentation (2023): Cloud and AI best practices.
- Hugging Face (2023): Whisper and BERT model specifications.
- MDN (2023): Web Audio API and WebSocket documentation.
- CrossWire Bible Society (2023): Public-domain Bible texts (KJV, WEB).
- U.S. Copyright Office (2023): Public domain guidelines.
- Librosa Documentation (2023): Audio preprocessing techniques.
- ONNX Runtime (2023): Model optimization for inference.

12. Conclusion

The VerseProjection MVP delivers real-time Bible verse auto-detection and projection, continuously analyzing preacher speech to map explicit or paraphrased quotes to public-domain KJV/WEB texts. The app outputs the top 10 matching verses for user selection, projects the chosen verse, and learns from selections to improve accuracy (5% increase in 3 months). The technical architecture (Web Audio API → Whisper → BERT → PostgreSQL → React, hosted on AWS) ensures <2-second latency, 90%+ accuracy, and scalability for 1,000–10,000 users. An offline Electron app supports low-internet environments. By focusing on verses only, the MVP avoids licensing complexities, staying within a $150K–$400K budget and 6–9-month timeline. The solution targets 10,000 churches ($2M revenue), addressing the need for automated, user-friendly verse projection during sermons.