# Frontend Specification - Debate Recording & Feedback Platform

## Overview

iOS/iPadOS application for recording debate speeches with integrated timing, automatic transcription, and AI-powered feedback generation.

## 1. User Roles & Authentication

### 1.1 Authenticated Mode

* **Target Users**: 6 teachers + 1 admin
* **Authentication**: Device-based authentication (phone/iPad specific)
* **Features Available**:
  + Auto-populated student lists based on schedule
  + Feedback history access
  + Motion suggestions based on schedule
  + Cloud sync of all debates and feedback
  + Full access to admin dashboard (admin only)

### 1.2 Guest Mode

* **Authentication**: None required
* **Features Available**:
  + Manual student list entry
  + Manual motion entry
  + Basic timer and recording functionality
  + Feedback available until next recording session starts
  + No history or cloud storage
* **Limitations**: Feedback link expires when new debate starts

## 2. Core User Flow

### 2.1 Pre-Debate Setup

#### Step 1: Session Type Selection

* Authenticated or Guest mode login

#### Step 2: Auto-Population (Authenticated Mode Only)

* App reads current time from device
* Queries schedule database via backend API
* Auto-populates:
  + Student list for current class
  + Suggested motion (if scheduled)
  + Default format and timings
* Teacher can:
  + Add additional students
  + Remove students (mark as absent)
  + Override motion
  + Adjust timings

#### Step 3: Manual Setup (Guest Mode or Override)

Fields to input:

- Student names (add/remove dynamically)

- Debate motion

- Debate format (Modified WSDC 3v3, BP 4x2, AP 2v2, Australs 3v3)

- Speech time limits

- Reply speech time (if applicable)

- Student level: Primary or Secondary (affects feedback template)

#### Step 4: Team Assignment

* Drag & drop interface
* **3v3 Format**: Drag students into Prop or Opp columns
* **BP Format**: Drag into OG, OO, CG, CO columns
* Visual confirmation of team composition
* Position assignment (Speaker 1, 2, 3 automatically assigned by drop order)

### 2.2 Debate Timer & Recording Interface

#### Main Timer Screen

Layout:

┌─────────────────────────────────────┐

│ Motion: [Display motion text] │

├─────────────────────────────────────┤

│ Current Speaker: │

│ [Name] - [Team] - [Position] │

├─────────────────────────────────────┤

│ │

│ [00:00] │

│ (Large Timer) │

│ │

│ 🔴 REC (indicator when active) │

│ │

├─────────────────────────────────────┤

│ [START/STOP Button] │

│ │

│ [< Previous] [Next Speaker >] │

│ │

│ Progress: Speaker 2/6 │

└─────────────────────────────────────┘

#### Timer Behavior

* **START pressed**:
  + Timer begins counting up
  + Audio recording starts automatically in background
  + REC indicator lights up
* **Bell Notifications**:
  + At **1:00** → 1 bell (ding)
  + At **[Time - 1:00]** → 1 bell (e.g., at 4:00 for 5-min speech)
  + At **[Time]** → 2 bells (ding-ding)
  + Every **15 seconds after** → 3 bells (ding-ding-ding)
* **STOP pressed** or **Swipe Right**:
  + Timer stops
  + Recording stops
  + Audio file saved locally with naming: {debate\_id}\_{speaker\_name}\_{position}\_{timestamp}.m4a
  + Upload to backend initiated immediately
  + Progress indicator shows upload status
  + Move to next speaker automatically
* **Swipe Left**: Go back to previous speaker (if needed)

#### Background Recording Requirements

* **Format**: M4A or AAC (compressed, good quality)
* **Sample Rate**: 44.1kHz or 48kHz
* **Channels**: Mono (sufficient for speech)
* **Quality**: Balance between file size and clarity (suggest 128kbps)

### 2.3 Real-time Upload & Processing

#### Upload Logic

On STOP button press:

1. Finalize audio recording

2. Save file locally with metadata:

{

debate\_id: UUID,

speaker\_name: string,

speaker\_position: string (e.g., "Prop 1", "OG 1"),

motion: string,

duration\_seconds: number,

timestamp: ISO8601,

student\_level: "primary" | "secondary"

}

3. Start upload to backend endpoint: POST /api/debates/{debate\_id}/speeches

4. Show upload progress (0-100%)

5. On success: Mark as uploaded, show checkmark

6. On failure: Retry 3 times, then alert user

7. Keep local file until confirmed processed by backend

#### Offline Handling (Future - Not Current Version)

* Queue uploads when offline
* Auto-upload when connection restored

### 2.4 Feedback Viewing

#### Access Points

* **During Debate**: "View Feedback" button available once any speech is processed
* **After Debate**: "View All Feedback" button
* **History**: "Past Debates" tab (authenticated mode only)

#### Feedback Display

┌─────────────────────────────────────┐

│ Speaker: [Name] │

│ Position: [Prop 1] │

│ Motion: [Motion text] │

├─────────────────────────────────────┤

│ Processing Status: ✓ Complete │

│ │

│ [View Feedback Doc] │

│ [Share Link] │

└─────────────────────────────────────┘

* Tap "View Feedback Doc" → Opens Google Docs link in web view or external browser
* "Share Link" → System share sheet with Google Docs URL

## 3. API Integration Requirements

### 3.1 Endpoints Needed

#### Authentication

POST /api/auth/login

Body: { teacher\_id: string, device\_id: string }

Response: { token: string, teacher: {...} }

#### Schedule & Auto-Population

GET /api/schedule/current?teacher\_id={id}&timestamp={ISO8601}

Response: {

class\_id: string,

students: [{ id, name, level }],

suggested\_motion: string,

format: string,

speech\_time: number

}

#### Debate Session

POST /api/debates/create

Body: {

teacher\_id: string,

motion: string,

format: string,

teams: {

prop: [student\_ids],

opp: [student\_ids]

},

speech\_time: number,

student\_level: "primary" | "secondary"

}

Response: { debate\_id: UUID }

#### Speech Upload

POST /api/debates/{debate\_id}/speeches

Content-Type: multipart/form-data

Fields:

- audio\_file: File

- speaker\_name: string

- speaker\_position: string

- duration\_seconds: number

- student\_level: string

Response: {

speech\_id: UUID,

status: "uploaded",

processing\_started: true

}

#### Feedback Status Check

GET /api/speeches/{speech\_id}/status

Response: {

status: "processing" | "complete" | "failed",

google\_doc\_url: string (if complete),

error\_message: string (if failed)

}

#### Feedback History

GET /api/teachers/{teacher\_id}/debates?limit=50

Response: {

debates: [{

debate\_id,

motion,

date,

speeches: [{

speaker\_name,

feedback\_url,

scores: {...}

}]

}]

}

## 4. UI/UX Design Guidelines

### 4.1 Color Scheme

* **Primary Action**: Blue (#007AFF - iOS blue)
* **Recording Active**: Red (#FF3B30)
* **Success**: Green (#34C759)
* **Prop Team**: Blue tint
* **Opp Team**: Red tint
* **BP Teams**: 4 distinct colors

### 4.2 Accessibility

* Large, tappable buttons (min 44x44pt)
* High contrast text
* VoiceOver support for timer announcements
* Haptic feedback on timer bells

### 4.3 Responsive Design

* iPhone: Single column layout
* iPad: Side-by-side team assignment, larger timer display

### 4.4 Critical User Feedback

* Upload progress indicators
* Processing status badges
* Error messages with retry options
* Network status indicator

## 5. Error Handling

### 5.1 Network Errors

Scenario: Upload fails

Action:

- Show retry button

- Auto-retry 3 times with exponential backoff

- Keep recording locally

- Alert user if all retries fail

### 5.2 Recording Errors

Scenario: Microphone permission denied

Action:

- Prompt user to enable in Settings

- Disable START button until permission granted

### 5.3 Backend Processing Errors

Scenario: Feedback generation fails

Action:

- Show "Processing failed" status

- Provide "Retry Processing" button

- Notify teacher via in-app notification

## 6. Data Models (Frontend)

### 6.1 Debate Session

interface DebateSession {

id: string;

motion: string;

format: "WSDC" | "BP" | "AP" | "Australs";

student\_level: "primary" | "secondary";

teams: {

prop?: Student[];

opp?: Student[];

og?: Student[];

oo?: Student[];

cg?: Student[];

co?: Student[];

};

speech\_time\_seconds: number;

reply\_time\_seconds?: number;

created\_at: string;

teacher\_id: string;

}

### 6.2 Speech Recording

interface SpeechRecording {

id: string;

debate\_id: string;

speaker\_name: string;

speaker\_position: string;

local\_file\_path: string;

duration\_seconds: number;

upload\_status: "pending" | "uploading" | "uploaded" | "failed";

processing\_status: "pending" | "processing" | "complete" | "failed";

feedback\_url?: string;

recorded\_at: string;

}

### 6.3 Student

interface Student {

id: string;

name: string;

level: "primary" | "secondary";

}

## 7. Performance Requirements

* **Timer Accuracy**: ±100ms
* **Recording Start Delay**: <200ms from START press
* **Upload Initiation**: <1s after STOP press
* **UI Responsiveness**: 60fps animations
* **App Launch**: <2s cold start

## 8. Key Implementation Notes for Developer

### 8.1 Recording While Timing

* Use AVAudioRecorder for background recording
* Start recording and timer simultaneously on START press
* Handle interruptions (phone calls, etc.) gracefully

### 8.2 File Naming Convention

{debate\_id}\_{speaker\_name\_sanitized}\_{position}\_{timestamp}.m4a

Example:

abc123\_john\_doe\_prop1\_20250120143045.m4a

### 8.3 Upload Strategy

* Use URLSession with background upload configuration
* Chunked upload for larger files (if speech >5min)
* Send metadata as JSON in multipart form

### 8.4 Bell Audio Files

* Provide 3 audio files:
  + bell\_1.mp3 (single ding)
  + bell\_2.mp3 (double ding)
  + bell\_3.mp3 (triple ding)
* Use AVAudioPlayer for precise timing

### 8.5 Gesture Recognition

* Swipe right: Next speaker
* Swipe left: Previous speaker (with confirmation)
* Tap timer: Pause/Resume (optional feature)

## 9. Future Enhancements (Not Current Version)

* Offline recording with queued uploads
* Student-facing app for viewing own feedback history
* Pattern analysis across multiple debates
* Real-time feedback preview (as speech happens)
* Video recording option
* Multi-language support
* Export feedback as PDF

## 10. Testing Checklist for Developer

* Timer accuracy across different speech lengths
* Bell notifications at correct intervals
* Simultaneous recording and timing
* Upload progress tracking
* Network failure recovery
* Background recording during interruptions
* Memory management for multiple recordings
* File cleanup after successful upload
* Google Docs link opening
* Share functionality
* Drag-drop team assignment
* Auto-population from schedule API
* Guest mode vs authenticated mode differences
* iPad and iPhone layouts

## 11. Dependencies & Third-Party Libraries

### Recommended Libraries

* **AVFoundation**: Audio recording and playback
* **Alamofire**: Network requests (or native URLSession)
* **SwiftUI/UIKit**: UI framework (your choice)
* **Combine**: Reactive programming for upload states

### API Integration

* Backend base URL: https://your-vps-domain.com/api (to be provided)
* Authentication: Bearer token in headers
* Request timeout: 30s (except uploads: 120s)

## End of Frontend Specification