SpeechWorks

Speech Therapy Treatment Tool

Technical Documentation

INEW 2434 - Advanced Web Programming

Ilhem Berzou

Houston City College

Fall 2025

Table of Contents

1. Introduction

1.1 Project Overview

SpeechWorks is a comprehensive web-based speech therapy treatment tool designed specifically for speech-language pathologists (SLPs). The application provides an integrated platform for managing clients, conducting therapy sessions, delivering interactive activities, and tracking progress over time.

The system addresses the need for a modern, technology-enhanced solution that can improve efficiency of therapy practices while maintaining detailed documentation capabilities.

1.2 Purpose and Scope

The primary purposes of SpeechWorks include:

- Client management with articulation, fluency, voice, and language profiles
- Treatment goal setting and tracking
- Session scheduling with SOAP notes documentation
- Interactive therapy activities with progress tracking
- Progress visualization with visual analytics
- Resource library for therapy materials

1.3 Technology Stack

SpeechWorks is built using the following technologies:

Layer	Technology
Frontend	AngularJS 1.8.x, HTML5, CSS3, JavaScript
Backend	Python FastAPI with SQLAlchemy ORM
Database	MySQL 8.0
Storage	AWS S3 for file uploads
Authentication	JWT (JSON Web Tokens) with bcrypt

2. System Architecture

2.1 Multi-Tier Architecture

SpeechWorks follows a three-tier architecture pattern that separates concerns and enhances scalability:

Presentation Layer (Frontend)

The frontend consists of a single-page application (SPA) that runs in the user's browser. It handles user interface rendering, input validation, and client-side state management. It communicates with the backend exclusively through RESTful API calls.

Application Layer (Backend)

The backend is a Python FastAPI application that contains all business logic, authorization, data validation, and API endpoints. The main.py file initializes the FastAPI application, configures CORS middleware for cross-origin requests, and registers all routers on startup.

Data Layer (Database)

MySQL database stores all persistent data including user accounts, client records, sessions, progress data, and resource metadata. SQLAlchemy ORM provides database abstraction.

2.2 Backend Structure

The backend is organized into the following modules:

Module	Purpose
`app/main.py`	Application entry point, CORS config
`app/routers/`	API route handlers (auth, clients, sessions, etc.)
`app/models/`	SQLAlchemy database models
`app/schemas/`	Pydantic validation schemas
`app/services/`	Business logic (auth_service, s3_service)

2.3 Frontend Structure

The frontend follows AngularJS MVC pattern:

File/Folder	Purpose
`index.html`	Main HTML template with all views
`js/app.js`	Angular module, services (ApiService, ToastService)

File/Folder	Purpose
`js/controllers/`	Controllers for each view (Dashboard, Clients, Sessions, Activities, Resources)
`css/styles.css`	Custom CSS with CSS variables, responsive design

3. Database Design

3.1 Entity Relationship Diagram

The database follows a relational model with the following key relationships:

- **Users → Clients:** One-to-many (therapist has many clients)
- **Clients → Goals:** One-to-many (client has many treatment goals)
- **Clients → Sessions:** One-to-many (client has many therapy sessions)
- **Sessions → SessionActivities:** One-to-many (session has many assigned activities)
- **Clients → ProgressRecords:** One-to-many (client has many progress records)

3.2 Core Tables

Users Table

Column	Type	Description
id	Integer PK	Auto-increment primary key
email	String(255)	Unique login email
hashed_password	String(255)	Bcrypt hashed password
full_name	String(255)	Therapist's full name
license_number	String(100)	Professional license number

Clients Table

Column	Type	Description
id	Integer PK	Auto-increment primary key
therapist_id	Integer FK	Reference to users table
first_name, last_name	String(100)	Client name
date_of_birth	Date	Client DOB
diagnosis	Text	Speech/language diagnosis
is_active	Boolean	Active/inactive status

Sessions Table

Column	Type	Description
id	Integer PK	Auto-increment primary key
client_id	Integer FK	Reference to clients table
session_date	DateTime	Scheduled date/time
duration_minutes	Integer	Session length (15-90 min)
status	Enum	scheduled, in_progress, completed, cancelled
soap_*	Text	SOAP notes (subjective, objective, assessment, plan)

4. API Documentation

4.1 Authentication Endpoints

Method	Endpoint	Description
POST	`/api/auth/register`	Create new therapist account
POST	`/api/auth/login`	Authenticate and get JWT token
GET	`/api/auth/me`	Get current user info

4.2 Client Endpoints

Method	Endpoint	Description
GET	`/api/clients`	List all clients
POST	`/api/clients`	Create new client
GET	`/api/clients/{id}`	Get client details with goals
PUT	`/api/clients/{id}`	Update client information

4.3 Session Endpoints

Method	Endpoint	Description
GET	`/api/sessions`	List sessions with filters
POST	`/api/sessions`	Create new session
PUT	`/api/sessions/{id}`	Update session/SOAP notes
GET	`/api/sessions/{id}/activities`	Get session activities
PUT	`/api/sessions/{id}/activities/{aid}`	Update activity record

5. Frontend Application

5.1 Controllers Overview

MainController

Handles authentication state, navigation, and global functions. Manages login/logout, view switching, and toast notifications.

DashboardController

Displays summary statistics, today's sessions (sorted by proximity to current time), quick actions, and recent activity.

ClientsController

Manages client CRUD operations, treatment goals, progress history, and client detail views with tabs for info, goals, progress, and history.

SessionsController

Handles session scheduling, status management, SOAP notes, activity assignment, and navigation to activities. Includes time-based validation for status changes.

ActivitiesController

Manages interactive therapy activities, trial tracking, progress recording, and return navigation to sessions after completion.

ResourcesController

Handles resource library with file uploads to S3, categorization, and search functionality.

6. Session Activity Workflow

The session activity workflow allows therapists to seamlessly navigate between session management and therapy activities with automatic data synchronization.

6.1 Clickable Activities on Session Cards

Activity tags displayed on session cards are interactive based on session status:

- **In-Progress Sessions:** Activities are clickable if not yet completed (trials_attempted = 0)
- **Completed Activities:** Display green badge with checkmark and accuracy percentage (e.g., "✓ 85%")
- **Other Statuses:** Activities displayed as read-only tags with tooltip "Session must be In Progress to start"
- **Click Hint:** Shows "Click an activity to start" when incomplete activities exist

6.2 Activity Navigation Flow

When a user clicks an activity tag on an in-progress session:

1. System validates session is "in_progress" and activity not completed
2. Client is retrieved from session data
3. Pending activity data stored on $rootScope.pendingActivity
4. Navigation triggers to Activities tab
5. ActivitiesController detects pending activity and auto-starts session with preselected client

6.3 Return Navigation After Completion

After activity completion:

- Progress record created via POST /api/progress
- Session activity record updated with trials_attempted, trials_correct, and accuracy
- $rootScope.returnToSessionId set to trigger session reload
- Automatic navigation back to Sessions tab
- Session card updates to show completed activity with accuracy badge

7. Session Status Management

7.1 Status Values

Status	Description
scheduled	Session is scheduled for future date/time
in_progress	Session is currently active, activities can be started
completed	Session finished, all activities completed
cancelled	Session was cancelled (displayed grayed out)

7.2 Time-Based Status Validation

Manual status change to "in_progress" is validated against session time window:

- **Session Window:** session_date to (session_date + duration_minutes)
- **Validation:** Current time must be within session window
- **Error:** "Cannot start session - session time (HH:MM) does not match current time"

7.3 Auto-Complete Sessions

Sessions automatically complete when all assigned activities are done:

- **Trigger:** Immediately after session activities load (checkSessionAutoComplete function)
- **Condition:** All assigned activities have trials_attempted > 0
- **Result:** Status updated to "completed" with toast notification

7.4 Timezone Handling

Session dates use convertLocalDateTimeToISO() function which creates ISO strings with timezone offset instead of UTC conversion, ensuring times display correctly in user's local timezone.

8. UI Components

8.1 Session Card Actions

Each session card displays action buttons with labels (hidden on mobile < 768px):

Button	Label	Function
Notes Button	Notes & Activities	Opens SOAP notes modal
Edit Button	Edit Session	Opens session edit modal
Delete Button	Cancel	Cancels session

8.2 Cancelled Session Styling

Location	Styling
Sessions Tab Cards	60% opacity, gray background, gray left border
Dashboard Today's Sessions	50% opacity, gray background, gray text

8.3 Activity Completion Badge

Completed activities display with visual indicators:

- **Background:** Green (#D1FAE5) with green border
- **Badge:** Dark green background with "✓ XX%" showing accuracy
- **Interaction:** Not clickable, cursor default

9. AWS Cloud Integration

9.1 AWS Services Used

Service	Purpose
Amazon RDS	MySQL database hosting (db.t3.micro)
Amazon S3	File storage for therapy resources and uploads
Amazon EC2	Backend API hosting (optional)

9.2 S3 Configuration

The S3 service (s3_service.py) handles:

- Presigned URL generation for secure uploads
- File categorization by resource type
- Unique filename generation to prevent conflicts

9.3 Estimated Costs

Service	Monthly Cost
RDS (db.t3.micro)	$15-25 (free tier eligible)
S3 Storage	$1-5 (depends on usage)
EC2 (if used)	$0 (750 hours/month free tier)

10. Deployment Guide

10.1 Backend Deployment

1. Clone repository and navigate to backend directory
2. Create virtual environment: python -m venv venv
3. Install dependencies: pip install -r requirements.txt
4. Configure .env file with database and S3 credentials
5. Run migrations to create database tables
6. Start server: uvicorn app.main:app --host 0.0.0.0 --port 8000

10.2 Frontend Deployment

1. Update API_BASE_URL in js/app.js to production backend URL
2. Deploy static files to web server (Nginx, Netlify, etc.)
3. Configure CORS on backend to allow frontend domain

10.3 Environment Variables

Variable	Description
`DATABASE_URL`	MySQL connection string
`SECRET_KEY`	JWT signing secret
`AWS_ACCESS_KEY_ID`	AWS credentials for S3
`AWS_SECRET_ACCESS_KEY`	AWS credentials for S3
`S3_BUCKET_NAME`	S3 bucket for file uploads

11. User Guide

11.1 Getting Started

1. Register a new account with your email and professional license number
2. Log in to access the dashboard
3. Add your first client with their information and diagnosis
4. Create treatment goals for the client
5. Schedule therapy sessions

11.2 Conducting Sessions

1. Navigate to Sessions tab
2. Change session status to "In Progress" when ready to start
3. Click on activity tags to start interactive therapy activities
4. Record correct/incorrect responses during activities
5. Session auto-completes when all activities are finished
6. Complete SOAP notes documentation

11.3 Tracking Progress

- View client progress in the Clients tab under Progress section
- Check activity history showing accuracy trends over time
- Update goal status as client progresses (In Progress, Mastered, etc.)

12. Implementation Plan

12.1 Development Timeline

Phase	Tasks
Week 1-2	Project setup, database design, authentication system
Week 3-4	Client management, goal tracking, session scheduling
Week 5-6	Interactive activities, progress tracking, SOAP notes
Week 7-8	Session-activity workflow, UI refinements, testing
Week 9-10	AWS integration, deployment, documentation

12.2 Future Enhancements

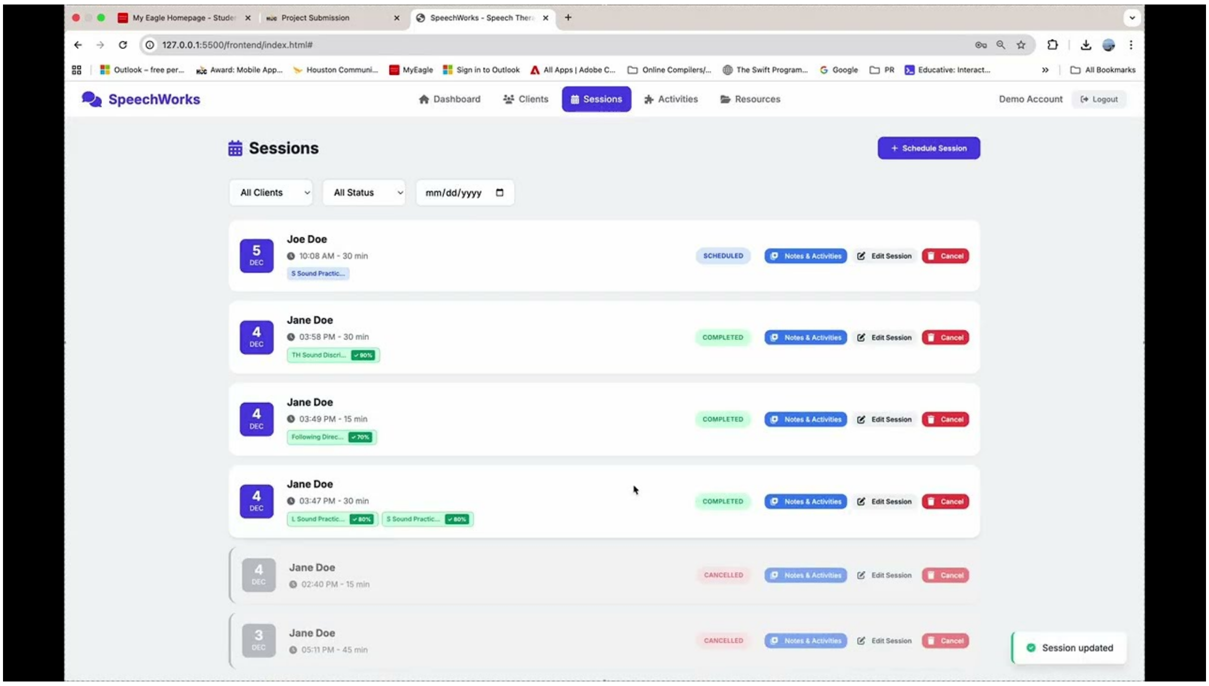
- Mobile-responsive native app version
- Video recording integration for session documentation
- Advanced analytics and reporting dashboards
- Multi-therapist practice management
- Parent/guardian portal for home practice
- Insurance billing integration

Conclusion

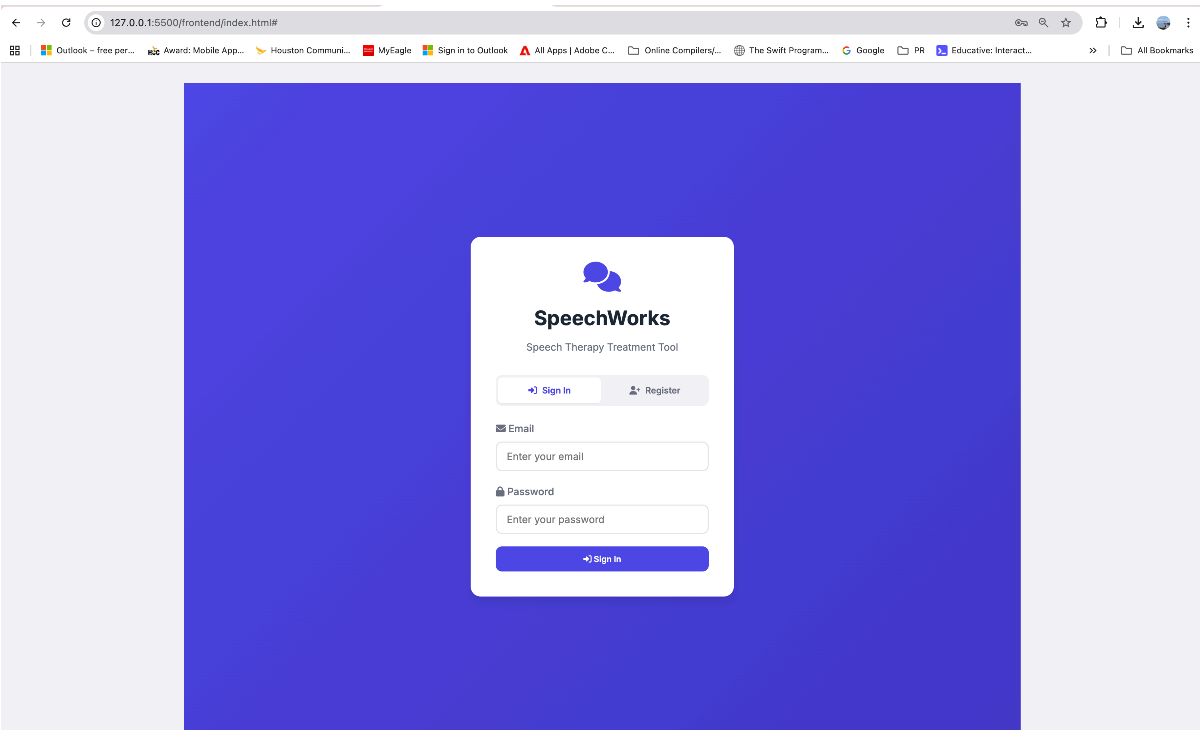
SpeechWorks provides a comprehensive solution for speech-language pathologists to manage their practice efficiently. The application combines client management, session scheduling, interactive therapy activities, and progress tracking in a single integrated platform. The session-activity workflow enables seamless navigation between scheduling and therapy delivery, while automatic status management reduces administrative burden.

Built with modern web technologies (AngularJS frontend, Python FastAPI backend, MySQL database), the application is designed for scalability and maintainability. AWS cloud integration provides reliable hosting and file storage capabilities.

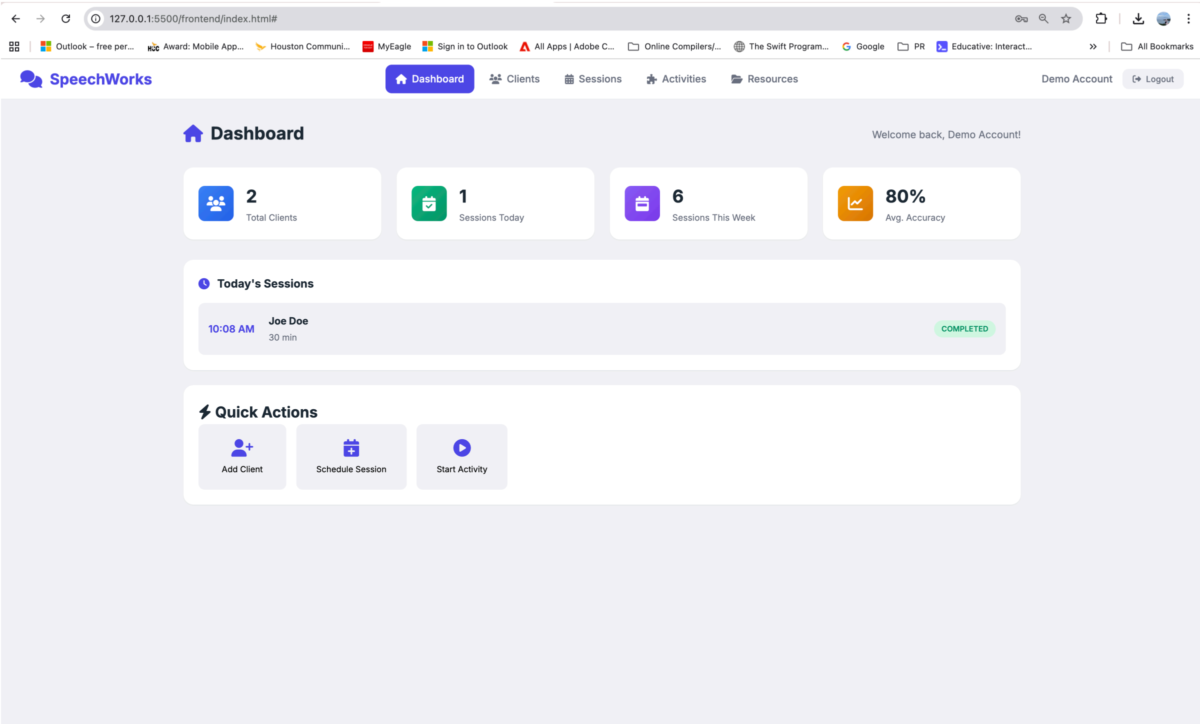
13. Video Demonstrating SpeechWorks App Working



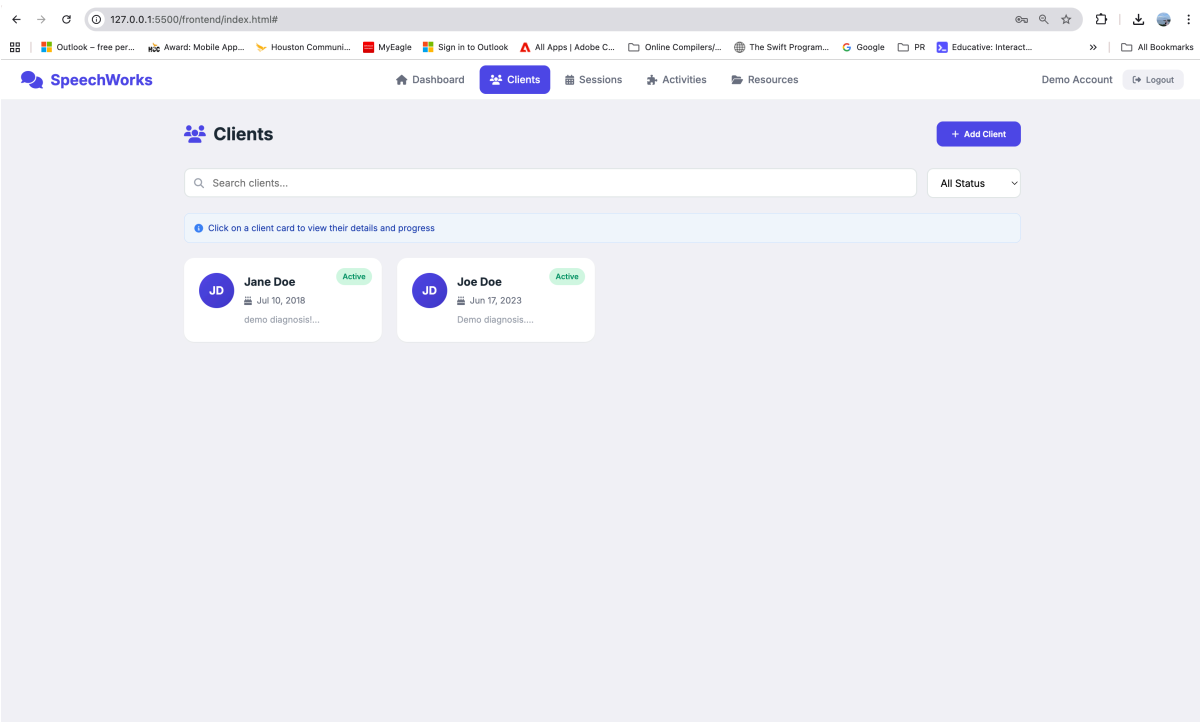
14. Screenshots of SpeechWorks App



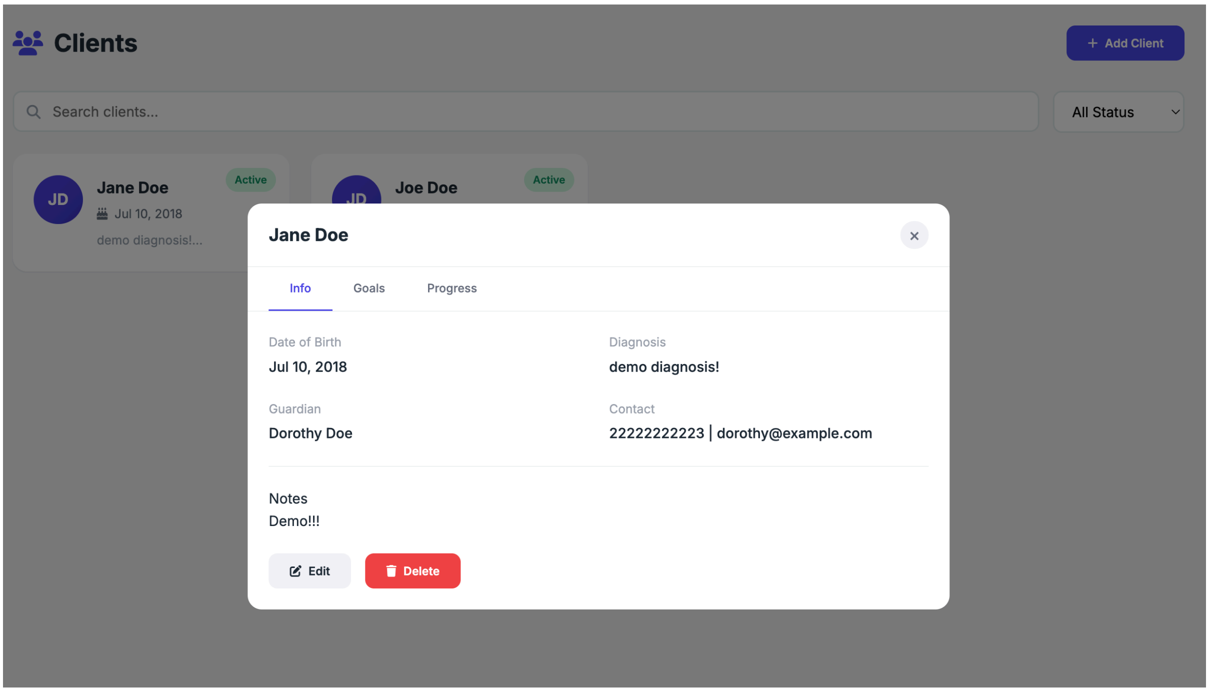
SpeechWorks Sign in/Registration page



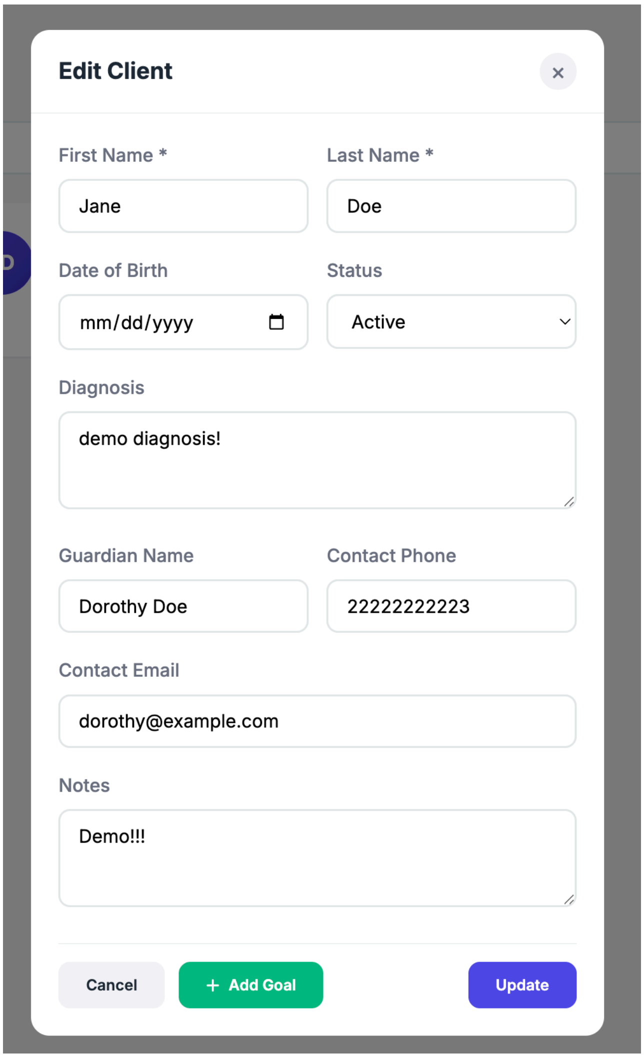
Landing Page After Login is Dashboard Tab



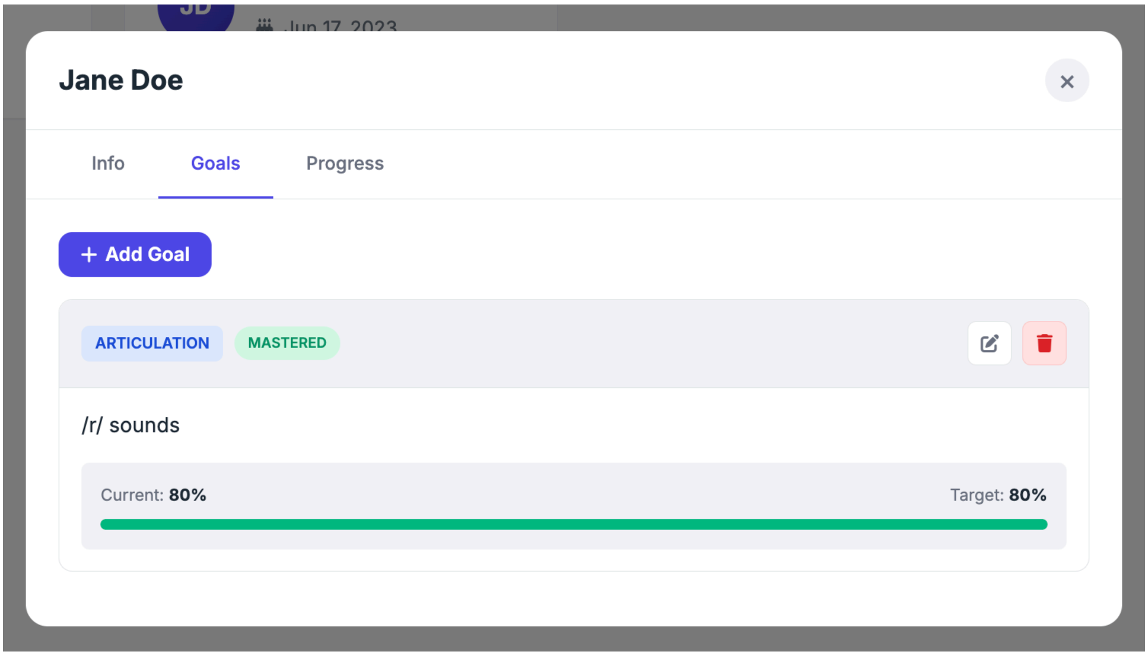
Clients Tab



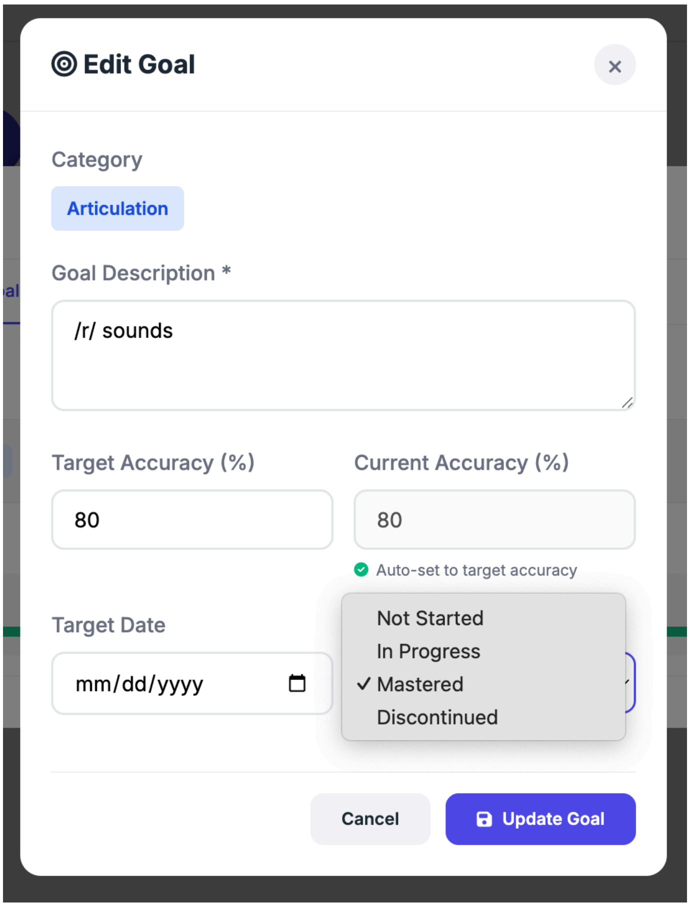
Client Card - Personal Info



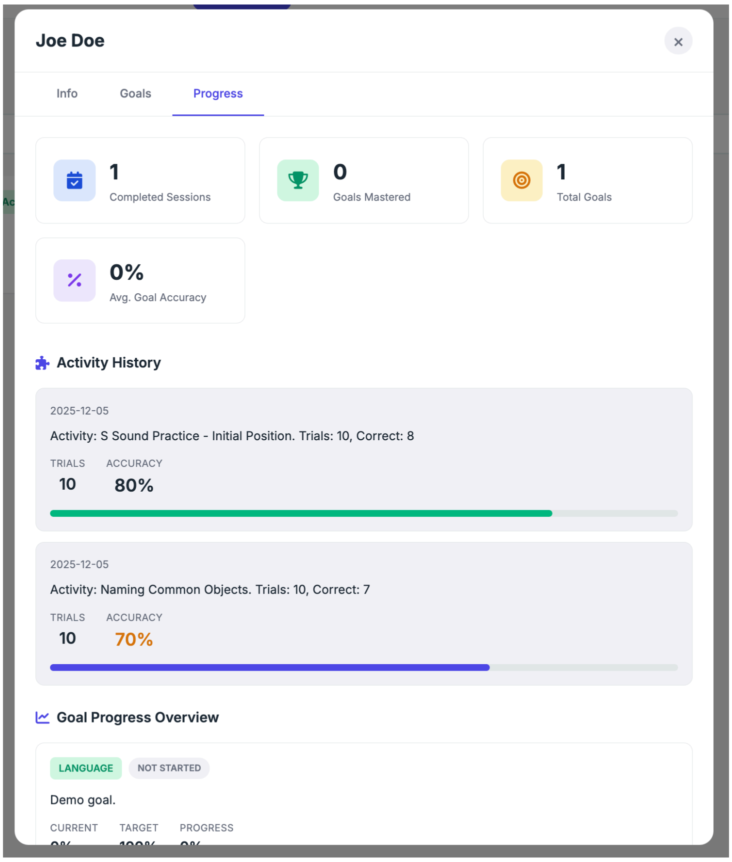
Edit Client Form



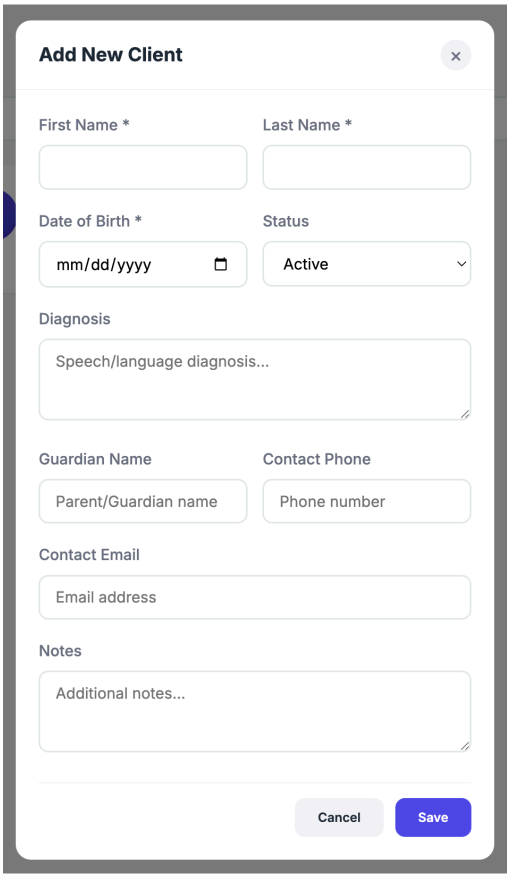
Client's Goals Tab



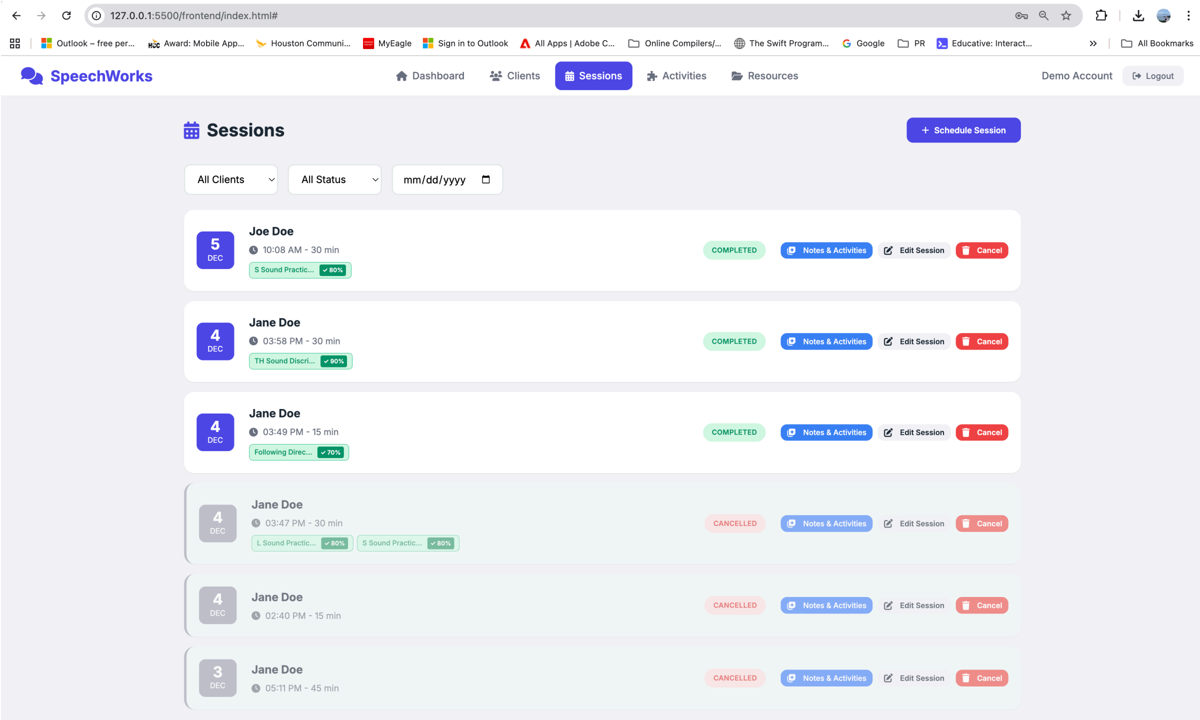
Edit Goal Form



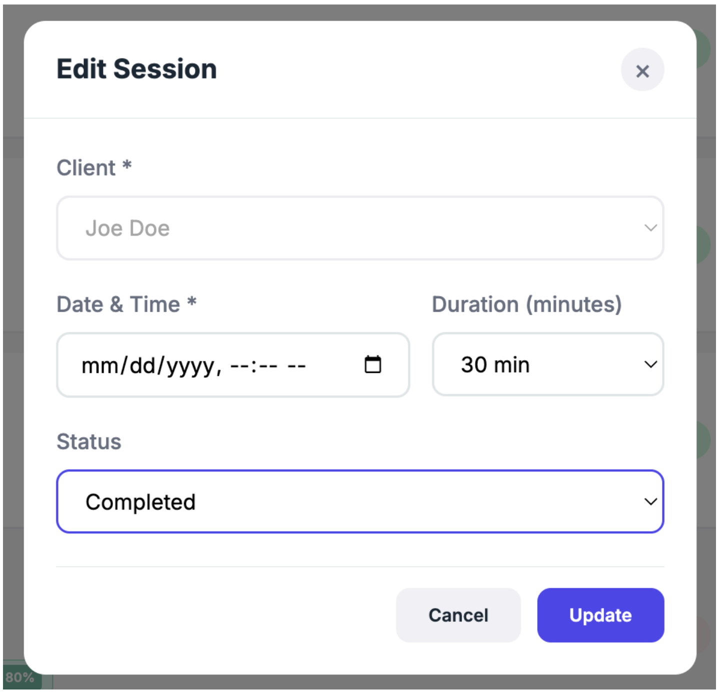
Client's Progress Tab



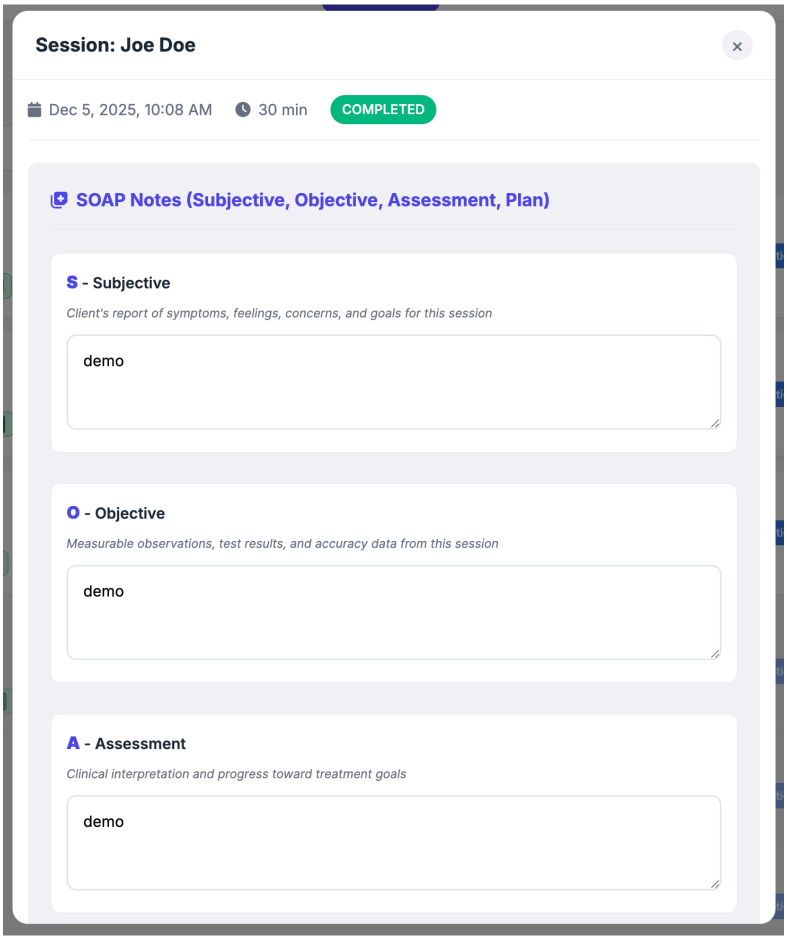
Add Client Form



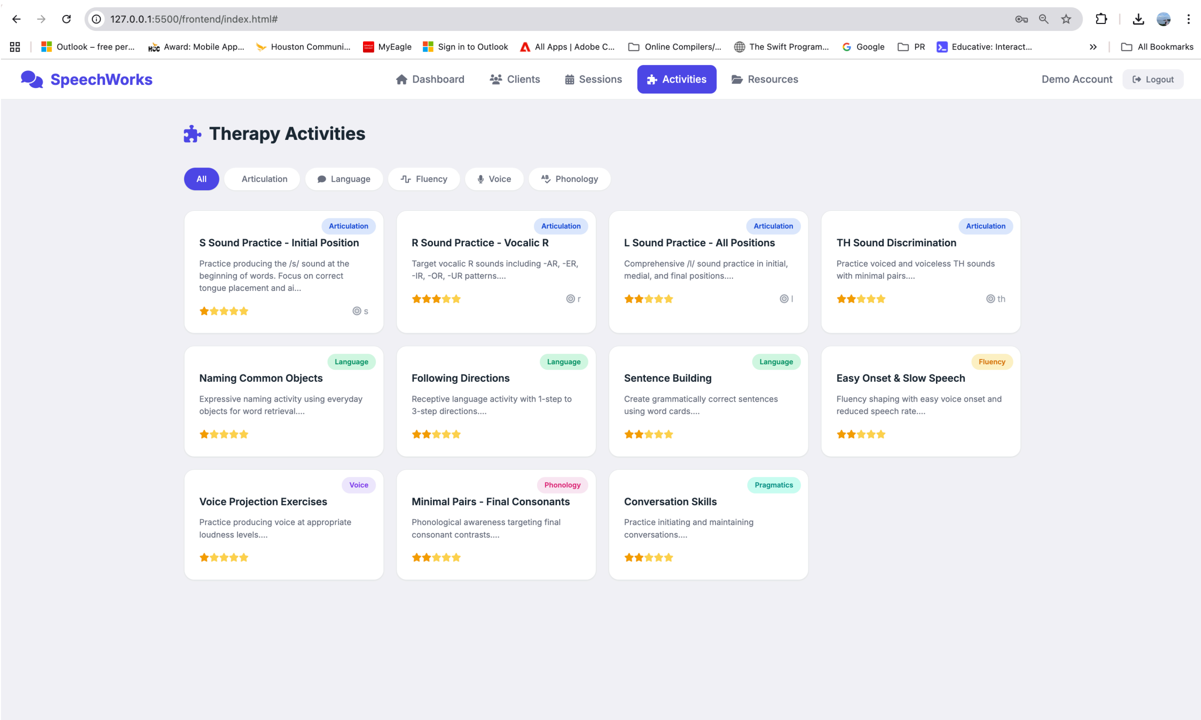
Sessions Tab



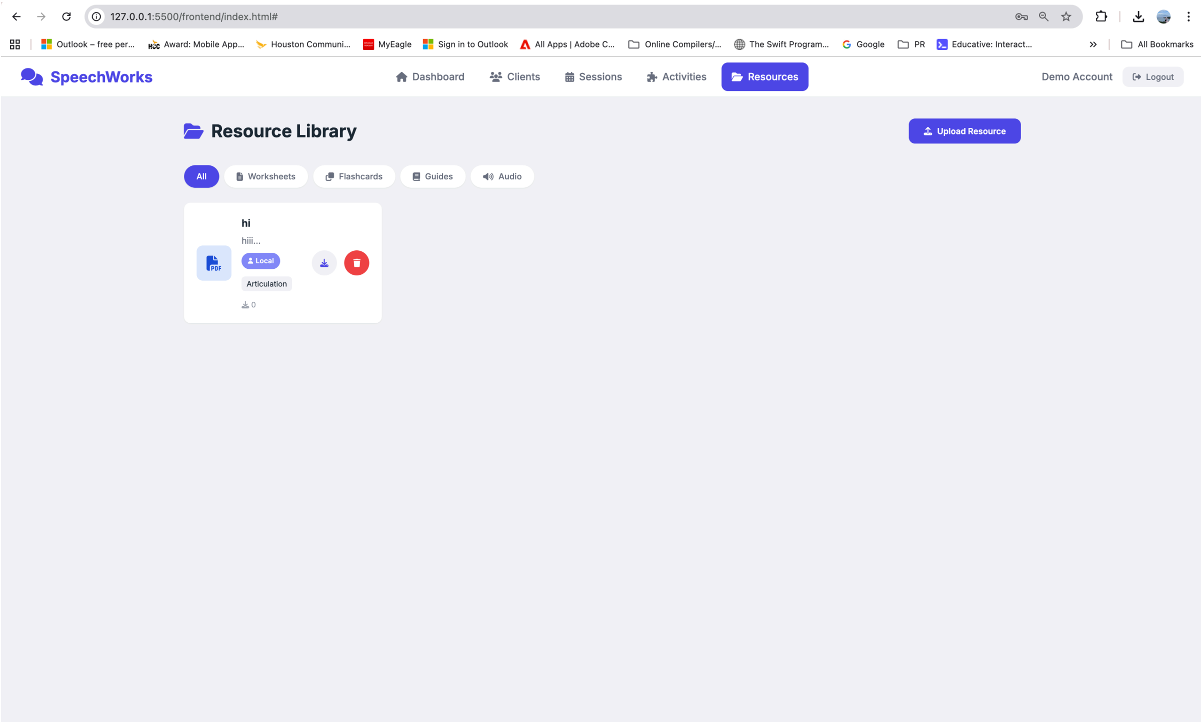
Edit Session Form



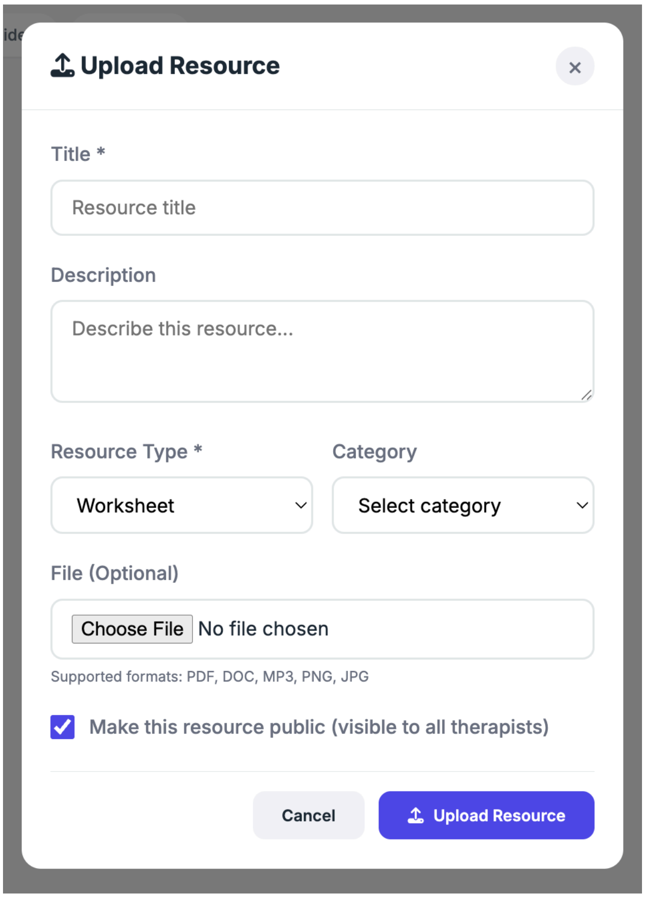
Session Notes Form



Activities Tab



Resources Tab



Upload Resource Form