API Endpoint Usage Documentation for SpeechSensePro (ToivoTek)

Authentication Endpoints

1. Login / Sign Up (Unified Endpoint)

This is a unified endpoint that handles both login and sign-up. If the user doesn't exist, a new account is created automatically.

Endpoint: `POST /api/auth/authenticate`

Description:

Initiates the authentication process by sending an OTP to the user's email. This endpoint works for both existing users (login) and new users (sign-up). If the email doesn't exist, a new user account is created with status `pending`.

Request Headers:

```
Content-Type: application/json
```

Request Body:

```
{

"turnstile_token": "string (required)",

"email": "string (required)"

}
```

Request Parameters:

- `turnstile_token` (string, required): Cloudflare Turnstile captcha token for bot protection
- `email` (string, required): User's email address

Response (Success - 200):

```
{
"success": true,
"token": "JWT_TOKEN_STRING",
"message": "OTP sent to your email"
}
```

Response Fields:

- `success` (boolean): Indicates if the request was successful
- `token` (string): JWT token for unverified user session. This token must be included in subsequent requests to verify the OTP
- `message` (string): Success message

Response (Error - 400):

```
{
"error": "turnstile_token and email are required"
}
```

or

```
{
"error": "Captcha verification failed. Please try again."
}
```

Response (Error - 500):

```
{
```

```
"error": "Internal server error"

}
```

Usage Example:

```
curl -X POST https://your-api-domain.com/api/auth/authenticate \

-H "Content-Type: application/json" \

-d '{

"turnstile_token": "0.abcdefghijklmnop",

"email": "user@example.com"

}'
```

Notes:

- The OTP is valid for 10 minutes
- The returned JWT token contains `sessionUuid` which is used to verify the OTP
- New users are created with `user_type: 'slp'` and `currentStatus: 'pending'`
- The OTP is automatically sent to the user's email address

---

2. OTP Verification

Endpoint: `POST /api/auth/verify`

Description:

Verifies the OTP code sent to the user's email. Upon successful verification, returns a JWT token that can be used for authenticated requests. The token type (verified/unverified) depends on the user's account status.

Request Headers:

```
Content-Type: application/json

Authorization: Bearer JWT_TOKEN_FROM_AUTHENTICATE
```

Request Body:

```
{

"otp": "string (required)"

}
```

Request Parameters:

- `otp` (string, required): The 6-digit OTP code received via email

Response (Success - 200):

For new users:

```
{

"success": true,

"isNew": true,

"name": null,

"token": "JWT_TOKEN_STRING",

"acceptedTos": false,

"acceptedAI": false,

"status": "unverified"

}
```

For active users:

```json
{

"success": true,

"name": "User Name",

"token": "JWT_TOKEN_STRING",

"isActive": true,

"status": "active",

"acceptedTos": true,

"acceptedAI": true,

"planName": "Premium Plan"

}
```

For existing unverified users:

```json
{

"success": true,

"isNew": false,

"name": "User Name",

"token": "JWT_TOKEN_STRING",

"acceptedTos": false,

"acceptedAI": false,

"status": "unverified",

"isActive": false

}
```

Response Fields:

- `success` (boolean): Indicates if the verification was successful
- `isNew` (boolean): Indicates if this is a new user account
- `name` (string|null): User's name from profile, if available
- `token` (string): JWT token for authenticated requests. Use this token in the `Authorization` header for subsequent API calls
- `acceptedTos` (boolean): Whether user has accepted Terms of Service
- `acceptedAI` (boolean): Whether user has accepted AI consent
- `status` (string): Current user status (`pending` , `unverified` , `reviewing` , `active` , etc.)
- `isActive` (boolean): Whether user has active status
- `planName` (string|undefined): Current subscription plan name (only for active users)

Response (Error - 400):

```
{
"error": "OTP is required"
}
```

or

```
{
"error": "Invalid or expired OTP"
}
```

Response (Error - 401):

```
{
"error": "No token provided"
}
```

or

```
{
"error": "Invalid token"
}
```

Response (Error - 404):

```
{
"error": "Session not found"
}
```

or

```
{
"error": "User not found"
}
```

Response (Error - 500):

```
{
"error": "Internal server error"
}
```

Usage Example:

```
curl -X POST https://your-api-domain.com/api/auth/verify \
-H "Content-Type: application/json" \

-H  "Authorization: Bearer JWT_TOKEN_FROM_AUTHENTICATE"  \

-d '{

"otp":  "123456"

}'
```

Notes:

- The OTP expires after 10 minutes
- The JWT token from `/authenticate` must be included in the Authorization header
- For active users, the token uses the user's UUID (verified token)
- For other statuses, the token uses the session UUID (unverified token)
- New SLP users with `pending` status are automatically changed to `unverified` after OTP verification

---

3. Verification Document Upload

Endpoint: `POST /api/auth/verification-upload`

Description:

Uploads a verification document (license, certificate, etc.) for account verification. The document is stored and the user's status is updated to `reviewing` pending admin approval.

Request Headers:

```
Authorization: Bearer JWT_TOKEN

Content-Type: multipart/form-data
```

Request Body (Form Data):

- `document` (file, required): Verification document file

- Allowed types: JPG, JPEG, PNG, WEBP, PDF
- Maximum file size: 10MB

Response (Success - 200):

```
{
"success": true,
"message": "Document uploaded successfully",
"filename": "uuid_timestamp.ext",
"status": "reviewing"
}
```

Response Fields:

- `success` (boolean): Indicates if the upload was successful
- `message` (string): Success message
- `filename` (string): The stored filename of the uploaded document
- `status` (string): Updated user status (will be `reviewing`)

Response (Error - 400):

```
{
"success": false,
"error": "No file uploaded or invalid file type. Only JPG, JPEG, PNG, WEBP, and PDF
}
```

Response (Error - 401):

```
{
```

```
"error": "No token provided"
}
```

or

```
{
"error": "Invalid token"
}
```

Response (Error - 404):

```
{
"error": "Session not found"
}
```

or

```
{
"error": "User not found"
}
```

Response (Error - 500):

```
{
"success": false,
"error": "File upload error"
}
```

or

```json
{
"error": "Internal server error"
}
```

Usage Example:

```bash
curl  -X  POST  https://your-api-domain.com/api/auth/verification-upload  \
-H "Authorization: Bearer JWT_TOKEN" \
-F  "document=@/path/to/document.pdf"
```

JavaScript/Fetch Example:

```javascript
const  formData = new  FormData();
formData.append('document', fileInput.files[0]);


fetch('https://your-api-domain.com/api/auth/verification-upload', {
method:  'POST',
headers: {
'Authorization':  `Bearer ${jwtToken}`
},
body:  formData
})
.then(response  =>  response.json())
.then(data  =>  console.log(data))
```

```
.catch(error => console.error('Error:', error));
```

Notes:

- The document is stored in `/var/www/controller/uploads/` directory
- Filename format: `{uuid}_{timestamp}.{extension}`
- User status is automatically updated to `reviewing` after successful upload
- The document filename is stored in the user's profile
- Only one document upload is tracked per user (subsequent uploads overwrite the previous one)

Authentication Flow Summary

Complete Flow for Email/OTP Authentication:

1. **Step 1: Authenticate (Login/Sign Up)**

- `POST /api/auth/authenticate`
- Send email and turnstile token
- Receive JWT token and OTP sent to email

2. **Step 2: Verify OTP**

- `POST /api/auth/verify`
- Send OTP code with JWT token from step 1
- Receive authenticated JWT token

3. **Step 3: Upload Verification Document (Optional)**

- `POST /api/auth/verification-upload`
- Upload verification document
- User status changes to `reviewing`

User Status Flow:

- `pending` → New user just created
- `unverified` → OTP verified, but no verification document uploaded
- `reviewing` → Verification document uploaded, awaiting admin approval
- `active` → Account fully verified and approved

Additional Authentication Endpoints

OAuth Authentication

Google OAuth

Endpoint: `POST /api/auth/google`

Request Body:

```
{
"code": "GOOGLE_ID_TOKEN"
}
```

Microsoft OAuth

Endpoint: `POST /api/auth/microsoft`

Request Body:

```
{
"code": "MICROSOFT_AUTHORIZATION_CODE"
}
```

Both OAuth endpoints follow a similar response structure to the `/verify` endpoint and return JWT tokens upon successful authentication.

Error Codes Reference

Status Code	Description
200	Success
400	Bad Request - Invalid input or expired OTP
401	Unauthorized - Missing or invalid token
403	Forbidden - Insufficient permissions
404	Not Found - Session or user not found
500	Internal Server Error

JWT Token Usage

After successful authentication, include the JWT token in subsequent API requests:

```
Authorization: Bearer YOUR_JWT_TOKEN
```

The JWT token contains:

- `userId` : User’s UUID
- `email` : User’s email address
- `userType` : User type (slp, admin, etc.)
- `status` : Current user status
- `acceptedTos` : Terms of service acceptance

- `acceptedAI` : AI consent acceptance
- `sessionUuid` : Session UUID (for unverified users)
- `authMethod` : Authentication method used
- `iat` : Token issued at timestamp