Image Upload Implementation Documentation

Overview

This document explains the image upload implementation in the HyperBuds frontend application, the current issue, and what the backend team needs to configure.

Current Implementation

Frontend Implementation

The frontend uses **FormData** to upload avatar images directly to the backend API endpoint. UploadThing has been removed and replaced with a native FormData-based solution.

Location: `src/lib/utils/uploadthing.ts`

How it works:

1. User selects an image file (JPG, PNG, GIF, or WebP)
2. Frontend validates file type and size (max 4MB)
3. Creates FormData with field name `"file"`
4. Sends POST request to `/api/v1/profiles/upload-media` with:
    - Authorization Bearer token in headers
    - FormData in request body
    - Content-Type automatically set by browser (multipart/form-data with boundary)

Request Format:

```
POST https://api-hyperbuds-backend.onrender.com/api/v1/profiles/upload-media
Headers:
  Authorization: Bearer <access_token>
  Content-Type: multipart/form-data; boundary=----WebKitFormBoundary...

Body (FormData):
  file: <File object>
```

Expected Response:
The backend should return a JSON response with the uploaded file URL in one of these formats:

- `{ url: "https://..." }`
- `{ data: { url: "https://..." } }`
- `"https://..."` (plain string)

Current Issue

Error: "Must supply api_key"

Error Details:

- **Status Code:** 500 Internal Server Error
- **Error Message:** `"Must supply api_key"`
- **Endpoint:** `POST /api/v1/profiles/upload-media`

What this means:
The backend is trying to upload the file to a storage service (likely Cloudinary, AWS S3, or similar) but doesn't have the API key configured in its environment variables.

Root Cause:
This is a **backend configuration issue**, not a frontend problem. The backend server needs to have the storage service API key configured in its environment variables.

What the Frontend Does

✅ What's Working

1. **File Validation:**
    - Validates file type (JPG, PNG, GIF, WebP only)
    - Validates file size (4MB maximum)
    - Provides clear error messages for invalid files
2. **Request Format:**
    - Creates FormData correctly with field name `"file"`
    - Includes authentication token in Authorization header
    - Sets proper Content-Type (handled automatically by browser)
3. **Error Handling:**
    - Catches and logs detailed error information

  - Provides user-friendly error messages
  - Logs full request/response details for debugging
4. **Response Parsing:**
  - Handles multiple response formats
  - Extracts URL from various response structures
  - Provides clear error if URL is missing

📝 Code Flow

```
uploadAvatar(file: File) →
  Validate file type/size →
  Create FormData with "file" field →
  POST to /profiles/upload-media →
  Parse response and return URL
```

What the Backend Team Needs to Do

Required Backend Configuration

The backend team needs to configure the storage service API key in the backend server's environment variables.

Steps for Backend Team:

1. **Identify the Storage Service:**
  - Check which storage service is being used (Cloudinary, AWS S3, Google Cloud Storage, etc.)
  - The error "Must supply api_key" typically comes from Cloudinary
2. **Get API Credentials:**
  - If using Cloudinary: Get API key and secret from Cloudinary dashboard
  - If using AWS S3: Get Access Key ID and Secret Access Key
  - If using another service: Get the appropriate API credentials
3. **Configure Environment Variables:**
   Add the API key to the backend server's environment variables. For example:
   If using Cloudinary:

   ```
   CLOUDINARY_API_KEY=your_api_key_here
   CLOUDINARY_API_SECRET=your_api_secret_here
   CLOUDINARY_CLOUD_NAME=your_cloud_name_here
   ```

If using AWS S3:

```
AWS_ACCESS_KEY_ID=your_access_key
AWS_SECRET_ACCESS_KEY=your_secret_key
AWS_S3_BUCKET_NAME=your_bucket_name
```

4. **Update Backend Code (if needed):**
   - Ensure the backend upload endpoint reads the API key from environment variables
   - Verify the backend uses the API key when uploading to the storage service
   - Test the endpoint after configuration
5. **Restart Backend Server:**
   - After adding environment variables, restart the backend server
   - Verify the environment variables are loaded correctly

Backend Endpoint Requirements

Endpoint: `POST /api/v1/profiles/upload-media`

Expected Request:

- Method: POST
- Content-Type: `multipart/form-data`
- FormData field name: `"file"`
- Headers: `Authorization: Bearer <token>`

Expected Response (Success):

```json
{
  "url": "https://storage-service-url.com/path/to/image.jpg"
}
```

OR

```json
{
  "data": {
    "url": "https://storage-service-url.com/path/to/image.jpg"
  }
}
```

Current Response (Error):

```
{
  "success": false,
  "message": "Error uploading media",
  "error": "Must supply api_key"
}
```

Testing

Frontend Testing

Once the backend is configured, test the upload functionality:

1. Navigate to Profile Edit page
2. Click “Upload Image” button
3. Select an image file (JPG, PNG, GIF, or WebP, max 4MB)
4. Verify the upload succeeds and the avatar updates

Backend Testing

The backend team can test the endpoint using curl:

```
curl -X POST https://api-hyperbuds-backend.onrender.com/api/v1/profiles/upload-media \
  -H "Authorization: Bearer <token>" \
  -F "file=@/path/to/image.jpg"
```

Expected response should include the uploaded file URL, not the “Must supply api_key” error.

Troubleshooting

If upload still fails after backend configuration:

1. **Check Backend Logs:**
   - Verify the API key is being read from environment variables
   - Check for any other errors in backend logs
2. **Verify Environment Variables:**
   - Ensure environment variables are set correctly
   - Verify the backend server has been restarted after adding variables

3. **Check Storage Service:**
    - Verify the storage service account is active
    - Check if API key has proper permissions
    - Verify storage service quota/limits
4. **Network Tab Inspection:**
    - Check the Network tab in browser DevTools
    - Verify the request is being sent correctly
    - Check the response status and body

Summary

- ✅ **Frontend:** Correctly implemented, sends FormData with “file” field
- ❌ **Backend:** Missing storage service API key configuration
- 🔧 **Solution:** Backend team needs to configure API key in environment variables
- 📝 **Status:** Frontend is ready, waiting for backend configuration

Contact

If you need more information or have questions about the frontend implementation, please contact the frontend team.

For backend configuration issues, contact the backend team with:

- Error message: “Must supply api_key”
- Endpoint: `/api/v1/profiles/upload-media`
- Request: FormData with “file” field