Software Test Design (STD)

1. Introduction:

The system processes raw 3D clothing files, primarily in .obj format, but also supports additional geometry formats when present in the dataset (e.g, pcd, mtl, png, etc). These files often contain multiple components such as displacement maps, metalness, normal maps, roughness, and opacity textures.

The system converts valid input files into the unified .glb format using trimesh, ensuring textures are preserved and visual fidelity is retained. All geometric and structural analyses—such as polygon count, bounding box volume, mesh density, and material detection—are computed and stored as metadata in a **PostgreSQL** database.

Meanwhile, the actual .glb files are stored in **MongoDB** using GridFS, enabling efficient storage and retrieval. The visual validation of converted models is supported through previewing in Blender, allowing human reviewers to ensure the converted assets appear as expected. This ensures that only fully valid, textured 3D garments are retained in the system.

Test Case ID	Description	Preconditions	Test Steps	Expected Result	Actual Result
TC-001	Upload a valid 3D model file	User is authenticated	1. Navigate to upload page 2. Select valid 3D file 3. Click 'Upload'	File is uploaded and saved in MongoDB	As expected, file is uploaded to GridFS
TC-002	Reject unsupported file format	User is authenticated	1. Navigate to upload page 2. Select .exe file 3. Click 'Upload'	System displays error message	System blocks file and shows 'unsupported format' warning
TC-003	Convert OBJ model to glTF	Valid OBJ file is uploaded	1. Run conversion module 2. Choose OBJ file 3. Wait for completion	glTF file is saved in output directory	glTF file successfully generated
TC-004	Store converted file in GridFS	glTF file generated	1. Load converted file 2. Save using GridFS API	File is accessible via GET route	GET /api/glb/<item_id> returned file
TC-005	Visualize uploaded model in browser	Valid glTF is stored in GridFS	1. Open visualization page 2. Load file by item_id	Model is displayed in Three.js viewer	Model renders correctly in browser
TC-006	Display metadata from PostgreSQL	Model entry exists in metadata table	1. Request item by ID 2. View metadata details	Model metadata shown in sidebar	Name, category, and polygon count displayed

TC-007	Reject upload without category tag	User is authenticated	1. Select file 2. Leave category empty 3. Submit	System shows validation error	Upload blocked, warning displayed
TC-008	Handle duplicate model upload	Model with same name exists	1. Try uploading file again 2. Observe response	System blocks or overwrites based on config	System overwrote file, metadata updated
TC-009	List all uploaded models	Multiple models uploaded	1. Navigate to list view 2. View available models	All model thumbnails are visible	List view correctly shows all entries
TC-010	Download converted file via API	glTF file exists in GridFS	1. Use download endpoint /api/glb/<item_id>	User receives the .glb file	File downloaded successfully with correct headers