Backend Implementation Plan

Phase 0 – Project Setup

1. **Initialize project**
   - npm init -y
   - Install dependencies:

npm install express mongoose jsonwebtoken bcryptjs multer aws-sdk socket.io cors helmet morgan

npm install --save-dev nodemon eslint

2. **Project structure**

```
backend/
├── src/
│   ├── config/      # env, DB, S3 config
│   ├── models/      # Mongoose models
│   ├── controllers/  # Route handlers
│   ├── routes/      # Express routers
│   ├── middleware/   # Auth, error handling
│   ├── services/     # Business logic
│   ├── utils/       # Helpers, constants
│   ├── sockets/     # WebSocket events
│   └── app.js       # Express app init
├── .env
├── package.json
└── server.js
```

Phase 1 – Core Setup

1. **Config**
   - .env variables: DB URL, JWT secret, AWS keys, etc.
   - config/db.js: connect to MongoDB.
   - config/s3.js: AWS S3 client instance.
2. **Security & Middleware**
   - CORS, Helmet for HTTP security.
   - Morgan for request logging.
   - Global error handler.

Phase 2 – Authentication & RBAC

1. **User Model**
   - Fields: tenantId, username, passwordHash, role, phaseId
   - Password hashing via bcryptjs.
2. **Auth Routes**

- POST /auth/register (Admin only)
- POST /auth/login
- GET /auth/me

3. **JWT Middleware**
   - Verify token, attach req.user.
4. **Role Middleware**
   - Check req.user.role & tenantId for resource access.

Phase 3 – Tenant & Phase Management

1. **Tenant Model**
   - Basic tenant info (multi-tenant support).
2. **Phase Model**
   - sequenceOrder, users[] linked to User IDs.
3. **Routes**
   - POST /admin/phase
   - PUT /admin/phase/:id
   - DELETE /admin/phase/:id

Phase 4 – Dynamic Form Builder

1. **ItemFormTemplate Model**
   - Stores array of field configs.
2. **Routes**
   - POST /admin/form-template — create/update
   - GET /admin/form-template
3. **Validation**
   - Middleware to validate incoming item data against form template.

Phase 5 – Item Management

1. **Item Model**
   - Tracking ID generator (6-digit, padded).
   - History array (phase, action, timestamp).
2. **Routes**
   - POST /items — single create
   - POST /items/bulk
   - PUT /items/:id/move-forward
   - PUT /items/bulk/move-forward
   - GET /items/:id/history
3. **Bulk Handling**
   - Accept CSV & manual ID list.

- Return partial success report.

Phase 6 – Return Workflow

1. **ReturnRequest Model**
   - Links fromPhaseId, toPhaseId, item list, status.
2. **Routes**
   - POST /returns
   - PUT /returns/:id/accept
   - PUT /returns/:id/reject
3. **Business Logic**
   - On accept → move items to target phase, update history.
   - On reject → log status.

Phase 7 – File Uploads

1. **Integration with S3**
   - Pre-signed URLs for secure uploads.
2. **Routes**
   - POST /files/presign → returns signed URL.
3. **Optional Processing**
   - Use Sharp or AWS Lambda for image resizing/compression.

Phase 8 – Real-Time Features

1. **Socket.io Setup**
   - Initialize in server.js, pass to routes.
2. **Events**
   - ITEM_MOVED, ITEM_RETURN_REQUEST, etc.
3. **Phase-based Channels**
   - Join rooms by phaseId & tenantId.
4. **SSE Support**
   - /dashboard/stream endpoint for read-only updates.

Phase 9 – Bulk Operation Reports

1. **Service**
   - On bulk op completion, generate CSV/JSON.
2. **Storage**
   - Save reports in S3 with signed download links.

Phase 10 – Deployment & Ops

1. **Dockerization**

- Multi-stage Dockerfile.

2. **Environments**
   - Dev, Staging, Prod with separate .env.
3. **Monitoring**
   - Winston logs, PM2 monitoring.
4. **Backups**
   - MongoDB dump cron jobs.

Implementation Order & Timeline

Week	Milestone
1	Project setup, DB config, basic auth
2	Tenant & Phase management
3	Form builder, item creation
4	Bulk operations, history tracking
5	Returns workflow
6	File uploads, S3 integration
7	Real-time updates (WebSocket/SSE)
8	Bulk reports, final testing
9	Deployment, monitoring setup