# Low Level Design Document

## Introduction

This Low Level Design (LLD) document outlines the implementation details for **MediTrack - Patient Record API**. The API provides secure CRUD operations for patient records, supports file uploads for medical reports, and enforces user authentication. The backend is built using Node.js, Express, and MongoDB.

---

## 1. System Components

| Component | Description | Key Responsibilities |
|---|---|---|
| API Server | Express-based HTTP server | Route requests, handle middleware |
| Auth Module | JWT-based authentication | User login, token validation |
| Patient Module | Patient record CRUD | Manage patient data |
| File Upload Module | Handles medical report uploads | Store/retrieve files |
| Database Layer | MongoDB via Mongoose | Data persistence |

---

## 2. Class/Interface Overview

| Class/Interface | Description | Key Methods/Attributes |
|---|---|---|
| `UserController` | Handles user auth | `login()` , `register()` |
| `PatientController` | CRUD for patients | `create()` , `read()` , `update()` , `delete()` |
| `FileController` | File upload/download | `uploadReport()` , `getReport()` |
| `AuthMiddleware` | Auth check | `verifyToken()` |
| `Patient` (Model) | Patient schema | `name` , `dob` , `reports[]` , etc. |
| `User` (Model) | User schema | `username` , `passwordHash` |

---

## 3. Data Structure Overview

| Model | Fields |
|---|---|
| User | `_id` , `username` , `passwordHash` |
| Patient | `_id` , `name` , `dob` , `gender` , `contactInfo` , `reports[]` |
| Report | `fileId` , `filename` , `uploadDate` , `uploadedBy` |

**Patient Schema Example:**

```
{
  _id: ObjectId,
  name: String,
  dob: Date,
  gender: String,
  contactInfo: { phone: String, email: String },
  reports: [{ fileId: ObjectId, filename: String, uploadDate: Date }]
}
```

---

## 4. Algorithms / Logic

**Authentication Flow (Pseudocode):**

```
POST /login:
  - Validate credentials
  - If valid, generate JWT token
  - Return token

Protected Route:
  - Extract token from header
  - Verify token
  - If valid, proceed; else, return 401
```

**File Upload Flow (Pseudocode):**

```
POST /patients/:id/reports:
  - Authenticate user
  - Validate file type/size
  - Store file (e.g., GridFS or local)
  - Update patient.reports[]
```

---

## 5. Error Handling

| Scenario | Handling/Response |
|---|---|
| Invalid credentials | 401 Unauthorized |
| Missing/invalid JWT | 401 Unauthorized |
| Resource not found (patient, file) | 404 Not Found |
| Validation error (input, file) | 400 Bad Request |
| File upload error | 500 Internal Server Error |
| Database error | 500 Internal Server Error |

---

**End of Document**