Low Level Design Document

Introduction

This Low Level Design (LLD) document outlines the core components and design for **PropList - Real Estate Listings API**. The API enables CRUD operations for property listings, image uploads, and user authentication, 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, signup, token verify
Listings Module	CRUD for property listings	Create, read, update, delete props
Image Upload	Handles image storage (local or cloud)	Upload, validate, link images
Database Layer	MongoDB via Mongoose	Data persistence, schema validation

---

2. Class/Interface Overview

Class/Interface	Description	Key Methods/Attributes
`UserController`	User auth endpoints	`signup()`, `login()`, `verifyToken()`
`ListingController`	Listing CRUD endpoints	`create()`, `getAll()`, `getById()`, `update()`, `delete()`
`ImageService`	Image upload logic	`uploadImage()`, `deleteImage()`
`User` (Model)	User schema/model	`email`, `passwordHash`, `role`
`Listing` (Model)	Listing schema/model	`title`, `desc`, `price`, `images`, `owner`

Relationships:

- `Listing` references `User` as `owner`.
- `Listing` contains array of image URLs.

---

3. Data Structure Overview

Model	Fields
User	`_id`, `email`, `passwordHash`, `role`
Listing	`_id`, `title`, `description`, `price`, `address`, `images` (array), `owner` (User ref)

Image	`url`, `filename`, `listingId` (optional)

---

4. Algorithms / Logic

Authentication Flow (Pseudocode):

```
POST /login:
  - Find user by email
  - Compare password hash
  - If valid, generate JWT token
  - Return token

Middleware verifyToken:
  - Extract JWT from header
  - Verify signature
  - Attach user to request or reject
```

Listing CRUD (Pseudocode):

```
POST /listings:
  - Auth middleware
  - Validate input
  - Save listing with owner=userId

GET /listings/:id:
  - Fetch listing by id
  - Return listing or 404
```

Image Upload (Pseudocode):

```
POST /listings/:id/images:
  - Auth middleware
  - Validate file type/size
  - Store file (local/cloud)
  - Update listing.images[]
```

---

5. Error Handling

Scenario	Handling Approach
Invalid credentials	401 Unauthorized, error message
Unauthorized access	403 Forbidden, error message
Resource not found	404 Not Found, error message
Validation errors	400 Bad Request, error details

File upload errors	400/500, error message
Database errors	500 Internal Server Error, log details

End of Document