a summary of the auth service verification and updates:

1. Directory Structure ✅

Code

CopyInsert

auth-service/

├── src/

│ ├── api/

│ │ ├── controllers/

│ │ │ └── auth.controller.js

│ │ ├── middleware/

│ │ │ ├── auth.middleware.js

│ │ │ ├── error-handler.js

│ │ │ ├── rate-limiter.js

│ │ │ └── validation.middleware.js

│ │ ├── routes/

│ │ │ └── auth.routes.js

│ │ └── validations/

│ │ └── auth.validation.js

│ ├── services/

│ │ ├── auth.service.js

│ │ └── redis.service.js

│ └── utils/

│ ├── errors.js

│ └── logger.js

1. Routes & Middleware Integration ✅

* All routes have proper middleware chains
* Consistent path imports
* Proper error handling
* Rate limiting on sensitive endpoints
* Validation schemas for all requests
* Authentication and role checks

1. Validation Schemas ✅ Created comprehensive validation schemas for:

* Login
* Token refresh
* Password reset/update
* MFA setup/verify
* All with proper error messages

1. Controller Updates ✅

* Consistent error handling
* Proper logging
* Standardized response format
* Clear separation of concerns
* Proper type checking

1. Middleware Updates ✅

* Authentication middleware with proper token validation
* Role-based authorization
* MFA verification
* Rate limiting with Redis integration
* Request validation with Joi
* Centralized error handling

1. Path Consistency ✅ Fixed all import paths to use consistent relative paths:

Code

CopyInsert

// Changed from

const { AuthError } = require('../../utils/errors');

// To

const { AuthError } = require('../utils/errors');

1. Error Handling ✅ Standardized error responses:

Code

CopyInsert

{

status: 'error',

code: 'ERROR\_CODE',

message: 'Error message',

details: [] // For validation errors

}

1. Logging ✅ Added comprehensive logging:

* Request context
* Error details
* User actions
* Security events

1. Security Features ✅

* Token validation
* Role-based access control
* MFA support
* Rate limiting
* Input validation
* Secure password handling

1. API Flow ✅ Each endpoint follows the pattern:
2. Route definition with middleware
3. Request validation
4. Controller processing
5. Service business logic
6. Standardized response