Library Management System - Backend Setup

1. Installing Required Packages

- bcrypt
- cloudinary
- cookie-parser
- cors
- dotenv
- express
- express-fileupload
- mongoose
- node-cron
- nodemailer

2. Setting Up the Project

1. Create a server.js file.
2. Modify package.json.
3. Create an app.js file and set up Express.
4. Create a config folder and add a config.env file with:
    - PORT
    - FRONTEND_URL
5. Configure the config.env file in app.js and set up necessary middlewares.
6. Set up server.js.

3. Database and Middleware Setup

1. Create a database folder with a db.js file for database connection and import it into app.js.
2. Create a middlewares folder with errorMiddleware.js and import it into app.js.
3. Add an error-handling (catchAsyncErrors) middleware to catch errors.

4. User Authentication System

1. Create a models folder and add userModel.js.
2. Create a controllers folder and add authController.js for authentication functions.
3. Implement a register function and add:
    - generateVerificationCode() in userModel.js.
    - sendVerificationCode() in utils/sendVerificationCode.js.
4. Create generateVerificationOtpEmailTemplate(verificationCode) in utils/emailTemplates.js.
5. Implement sendEmail() in utils/sendEmail.js and add the following environment variables to your config file.
    - SMTP_HOST = smtp.gmail.com
    - SMTP_SERVICE = gmail
    - SMTP_PORT = 465

- SMTP_MAIL
    - SMTP_PASSWORD

6. Create an authRoutes.js file in routes and add a register route.
7. Import and use this router in app.js.
8. Test the register route using Postman.

5. Authentication & Authorization

1. Add verifyOTP in authController.js.
2. Create sendToken() in utils/sendToken.js and generateToken() in userModel.js and add the following environment variables to your config file.
    - JWT_SECRET_KEY
    - JWT_EXPIRE
    - COOKIE_EXPIRE
3. Test the verifyOTP route in Postman.
4. Implement login and logout routes and test them.
5. Create authMiddleware.js with isAuthenticated function.
6. Apply this middleware to the logout function and test it.
7. Add a getUser route and test it.

6. Password Management

1. Create a forgotPassword route and implement:
    - getResetPasswordToken() in userModel.js.
    - generateForgotPasswordEmailTemplate(resetPasswordUrl) in emailTemplates.js.
2. Test the forgotPassword route in Postman.
3. Implement the resetPassword and updatePassword routes and test them.

7. Book Management

1. Create bookModel.js in models and bookRoutes.js in routes.
2. Add an isAuthorized middleware in authMiddleware.js.
3. Create bookController.js and implement:

- addBook (Test on Postman)
- getAllBooks (Test on Postman)
- deleteBook (Test on Postman)

8. Borrowing System

1. Create borrowModel.js and borrowRoutes.js in routes.
2. Implement borrowController.js with:

- recordBorrowBook (Test on Postman)
- returnBorrowBook + calculateFine in utils/fineCalculator.js (Test on Postman)
- borrowedBooks and getBorrowedByAdmin (Test on Postman)

9. User Management

1. Create userController.js and implement:
2. - getAllUsers
3. - registerNewAdmin

4. Configure Cloudinary for media uploads. Also add the following environment variables to config file.
    - CLOUDINARY_CLIENT_NAME
    - CLOUDINARY_CLIENT_API
    - CLOUDINARY_CLIENT_SECRET

10. Automation Services

1. Create a services folder and add:
2. - notifyUsers.js (Automatically sends email reminders for due books).
3. - removeUnverifiedAccounts.js (Deletes unverified accounts automatically).

✅ Backend Setup Complete! 🎉