**User Routes**

This Node.js Express routing configuration file sets up various routes for user authentication and management. It uses the Express router to define endpoints for actions such as signing up, activating an account, signing in, and more. The routes are associated with corresponding controller functions from the 'userController' file. Additionally, there are routes for getting a single user and updating their password, both protected by the 'userAuthorize' middleware to ensure proper authorization. The file exports the configured router for use in other parts of the application.

**Admin Routes**

In summary, this file sets up routes for administrative actions using Express. The routes are protected by user and admin authorization middleware, ensuring that only authorized users and administrators can perform the specified actions. The actual functionality for updating passwords and managing environment variables is implemented in the 'adminController' file, which is imported and utilized in this routing configuration.

**Assignment Routes**

The router defines two routes: one for getting all assignments and one for adding an assignment.

Middleware functions such as adminAuthorize, instructorAuthorize, and userAuthorize are used to authorize access to the routes.

Only users with specific roles (admin, instructor) are allowed to add assignments.

Users with the roles of admin and user are required to authorize access to retrieve all assignments.

The getAllAssignments function is responsible for retrieving all assignments.

The addAssignment function is responsible for adding an assignment.

**Auth Routes**

The code snippet shows a router definition in an Express.js application for handling a POST request to the '/forgot-password' route.

Key Takeaways

The code makes use of the Express.js library to define routes.

The code exports the router instance for use in other parts of the application.

The router is defined to handle the POST request to the '/forgot-password' route.

The router delegates the handling of the request to the 'sendPasswordResetEmail' function defined in 'authController.js'.

The 'sendPasswordResetEmail' function is imported using the 'require' function from the '../controllers/authController' module.

The code suggests that the application has a separate module for handling authentication-related functionality.

The 'sendPasswordResetEmail' function is expected to implement the logic for sending a password reset email.

**Billing Address Routes**

This text is a code snippet that defines routes using the Express framework in a Node.js application. It imports necessary dependencies, initializes an Express Router object, and assigns middleware and controller functions to specific routes.

Key Takeaways

The code snippet is defining routes using the Express framework in a Node.js application.

The express module is imported and used to create an instance of the Router object.

Middleware functions like adminAuthorize and userAuthorize are imported and applied to certain routes.

Controller functions for storing, fetching, and updating billing addresses are imported from billingAddressController.js.

The routes /store, /fetch, and /update are defined and assigned the corresponding controller functions.

The router object is exported to be used by other parts of the application.

**Book Counseling Routes**

The text provides an example code snippet in JavaScript using the Express framework to create a router for handling book counselling endpoints, with authentication middleware for users and administrators.

Key Takeaways

The code snippet uses the Express framework for building a router in JavaScript.

Authentication middleware (userAuthorize and adminAuthorize) is utilized for user and administrator access control.

Two endpoints are defined: a GET endpoint (getAllBookCounselling) and a POST endpoint (bookAFreeCounselling).

The GET endpoint requires both user and admin authorization.

The POST endpoint does not have any specific authorization requirements.

The code exports the defined router for external use.

The code file likely exists in a directory structure where middleware and controllers are organized in separate folders.

**Certificate Routes**

This Express routing configuration sets up two routes for handling certificates:

Upload Certificate:

HTTP method: PATCH

Middleware: Requires user and instructor authorization.

Controller function: uploadCertificate

Endpoint: '/'

Download Certificate:

HTTP method: GET

Middleware: Requires user authorization.

Controller function: downloadCertificate

Endpoint: '/:id'

The file utilizes the 'express' module, defines routes for uploading and downloading certificates, and employs middleware for user and instructor authorization. The actual functionality for certificate upload and download is implemented in the 'certificateController', which is imported and linked to the respective routes.

**Coupon Code Routes**

This Express routing configuration sets up endpoints for managing coupon codes, with various actions accessible based on user roles:

Get All Coupon Codes:

HTTP method: GET

Middleware: Requires user authorization.

Controller function: getAllCouponCode

Endpoint: '/'

Create Coupon Code:

HTTP method: POST

Middleware: Requires user and admin authorization.

Controller function: createCouponCode

Endpoint: '/'

Apply Coupon Code:

HTTP method: POST

Middleware: Requires user authorization.

Controller function: applyCouponCode

Endpoint: '/apply'

Change Coupon Code Status:

HTTP method: PATCH

Middleware: Requires user and admin authorization.

Controller function: updateCouponStatus

Endpoint: '/:id'

Delete Coupon Code:

HTTP method: DELETE

Middleware: Requires user and admin authorization.

Controller function: deleteCouponCodeById

Endpoint: '/:id'

The file utilizes the 'express' module, defines routes for coupon code management, and includes middleware for user and admin authorization. The actual functionality for these actions is implemented in the 'couponCodeController', which is imported and linked to the respective routes.

**Course Enroll Route**

This Express routing configuration handles various actions related to course enrollment and order management:

Get Enrolled Courses for a Student:

HTTP method: GET

Middleware: Requires user authorization.

Controller function: getEnrolledCourse

Endpoint: '/student'

Get Enrolled and Not Refunded Courses for a Student:

HTTP method: GET

Middleware: Requires user authorization.

Controller function: getEnrolledAndNotRefundCourse

Endpoint: '/student-not-refund'

Get Recent Enrolled Orders for Admin Dashboard:

HTTP method: GET

Middleware: Requires user and admin authorization.

Controller function: getRecentOrders

Endpoint: '/recent-orders'

Course Checkout in USD:

HTTP method: POST

Middleware: Requires user authorization.

Controller function: enrollCourseByUSD

Endpoint: '/enroll-in-usd'

Stripe Webhook:

HTTP method: POST

Controller function: postStripeWebHook

Endpoint: '/stripe/webhook'

Course Checkout in INR:

HTTP method: POST

Middleware: Requires user authorization.

Controller function: enrollCourseByINR

Endpoint: '/enroll-in-inr'

Razorpay Verification:

HTTP method: POST

Middleware: Requires user authorization.

Controller function: razorpayVerify

Endpoint: '/razorpay-verify'

Update Refund Request Status:

HTTP method: PATCH

Middleware: Requires user authorization.

Controller function: updateRefundRequest

Endpoint: '/update-refund-request/:id'

The file utilizes the 'express' module, defines routes for course enrollment and order management, and includes middleware for user and admin authorization. The actual functionality for these actions is implemented in the 'courseEnrollController', which is imported and linked to the respective routes.

**Course Route**

The provided Express routing configuration file manages various actions related to courses, including retrieval, addition, deletion, and video uploads. Key points include:

Get Top Sales Course:

HTTP method: GET

Middleware: Requires user and admin authorization.

Controller function: getTopSalesCourse

Endpoint: '/top-sale'

Get All Courses:

HTTP method: GET

Controller function: getAllCourses

Endpoint: '/'

Add Course:

HTTP method: POST

Middleware: Requires user and admin authorization.

Controller function: addCourseTitle

Endpoint: '/'

Upload Course Video:

HTTP method: POST

Middleware: Requires user and admin authorization.

Controller function: uploadCourseVideo

Endpoint: '/upload-video'

Get Course by ID:

HTTP method: GET

Controller function: getCourseById

Endpoint: '/:id'

Delete Course by ID:

HTTP method: DELETE

Middleware: Requires user and admin authorization.

Controller function: deleteCourseById

Endpoint: '/:id'

The file utilizes the 'express' module, defines routes for course-related actions, and includes middleware for user and admin authorization. The actual functionality for these actions is implemented in the 'courseController', which is imported and linked to the respective routes.

**Profile Route**

This Express routing configuration file handles various user profile-related actions, including fetching profiles, updating user information, and managing user status and deletion. Key points include:

Get User Profile:

HTTP method: GET

Middleware: Requires user authorization.

Controller function: getProfile

Endpoint: '/'

Update User Profile:

HTTP method: PATCH

Middleware: Requires user authorization.

Controller function: updateProfileById

Endpoint: '/:id'

Get All Students:

HTTP method: GET

Middleware: Requires user and admin authorization.

Controller function: getAllStudents

Endpoint: '/students'

Get All Users (with Pagination):

HTTP method: GET

Middleware: Requires user and admin authorization.

Controller function: getAllUsers

Endpoint: '/all-users'

Update User Status by ID:

HTTP method: PATCH

Middleware: Requires user and admin authorization.

Controller function: updateUserStatusById

Endpoint: '/status/:id'

Delete User by ID:

HTTP method: DELETE

Middleware: Requires user and admin authorization.

Controller function: deleteUserById

Endpoint: '/delete/:id'

The file utilizes the 'express' module, defines routes for user profile actions, and includes middleware for user and admin authorization. It also provides detailed API documentation using comments with API specifications for each route. The actual functionality for these actions is implemented in the 'profileController', which is imported and linked to the respective routes.

**Refund Terms Routes**

This Express routing configuration file manages refund terms and related actions, providing functionality for creating new terms, retrieving existing terms, and updating registration fees and return windows. Key points include:

Create Refund Terms:

HTTP method: POST

Middleware: Requires user and admin authorization.

Controller function: createRefundTerms

Endpoint: '/'

Get Refund Terms:

HTTP method: GET

Middleware: Requires user authorization.

Controller function: getRefundTerms

Endpoint: '/'

Update Registration Fees:

HTTP method: PUT

Middleware: Requires user and admin authorization.

Controller function: updateRegistrationFees

Endpoint: '/'

Update Return Window:

HTTP method: PATCH

Middleware: Requires user and admin authorization.

Controller function: updateReturnWindow

Endpoint: '/'

The file utilizes the 'express' module, defines routes for managing refund terms, and includes middleware for user and admin authorization. The actual functionality for these actions is implemented in the 'refundTermsController', which is imported and linked to the respective routes.

**Reward Route**

This Express routing configuration file manages reward-related actions, including retrieving all rewards, creating new rewards, redeeming reward points, and deleting rewards by ID. Key points include:

Get All Rewards:

HTTP method: GET

Middleware: Requires user authorization.

Controller function: getAllRewards

Endpoint: '/'

Create Reward:

HTTP method: POST

Middleware: Requires user authorization.

Controller function: createReward

Endpoint: '/'

Redeem Reward Points:

HTTP method: POST

Middleware: Requires user authorization.

Controller function: reedemRewardPoints

Endpoint: '/redeem'

Delete Reward By ID:

HTTP method: DELETE

Middleware: Requires user authorization.

Controller function: deleteRewardById

Endpoint: '/:id'

The file utilizes the 'express' module, defines routes for managing rewards, and includes middleware for user authorization. The actual functionality for these actions is implemented in the 'rewardController', which is imported and linked to the respective routes.

**User Routes**

This Express routing configuration file handles various user authentication and profile-related actions. Key points include:

User Authentication Routes:

Sign Up: POST request to '/sign-up' for user registration.

Activate Account: POST request to '/activate-account' for activating user accounts.

Resend Activation Code: POST request to '/resend-code' for resending activation codes.

Sign In: POST request to '/sign-in' for user login.

Google Sign In: POST request to '/google-sign-in' for user authentication via Google.

Forgot Password: POST request to '/forgot-password' for initiating password reset.

Reset Password: POST request to '/reset-password/:token' for resetting user passwords using a token.

Get Single User:

GET request to '/single' for retrieving details of a single user.

Requires user authorization middleware.

Update User Password:

POST request to '/update-password' for updating user passwords.

Requires user authorization middleware.

The file utilizes the 'express' module, defines routes for user authentication and profile actions, and includes middleware for user authorization. The actual functionality for these actions is implemented in the 'userController', which is imported and linked to the respective routes.