**Documentation for Account Schema and Company Routes**

**1. Account Schema**

**1.1 Overview**

The Account schema defines the structure for account-related data in the application, utilizing MongoDB and Mongoose. It includes details about the account, associated transactions, and fiscal year information.

**1.2 Schema Definition**

|  |
| --- |
| const mongoose = require('mongoose');  const openingBalanceByFiscalYearSchema = new mongoose.Schema({  date: {  type: Date,  default: () => new Date().toISOString // Sets the default date to the current date  },  amount: {  type: Number,  default: 0 // Default amount set to 0  },  type: {  type: String,  enum: ['Dr', 'Cr'], // Defines the type of balance (Debit or Credit)  default: 'Dr' // Default type set to Debit  },  fiscalYear: {  type: mongoose.Schema.Types.ObjectId,  ref: 'FiscalYear', // References the FiscalYear model  required: true // This field is required  }  });  // Main Account Schema  const accountSchema = new mongoose.Schema({  name: {  type: String,  required: true, // Account name is mandatory  },  address: String,  ward: Number,  phone: String,  pan: {  type: Number,  min: 9, // PAN must be a minimum of 9 digits  },  contactperson: String,  email: {  type: String,  },  openingBalanceByFiscalYear: [openingBalanceByFiscalYearSchema], // Stores opening balances for each fiscal year  openingBalance: {  fiscalYear: {  type: mongoose.Schema.Types.ObjectId,  ref: 'FiscalYear' // Reference to the current fiscal year  },  amount: {  type: Number,  default: 0 // Default opening balance amount  },  type: {  type: String,  enum: ['Dr', 'Cr'], // Defines if the balance is Debit or Credit  default: 'Dr' // Default balance type  }  },  openingBalanceDate: {  type: Date,  default: new Date('2023-07-17') // Default date for opening balance  },  companyGroups: {  type: mongoose.Schema.Types.ObjectId,  ref: 'CompanyGroup',  required: true // This field is required  },  date: {  type: Date,  default: Date.now // Default date set to current date  },  company: {  type: mongoose.Schema.Types.ObjectId,  ref: 'Company',  required: true // This field is required  },  transactions: [ // Array of references to Transaction documents  {  type: mongoose.Schema.Types.ObjectId,  ref: 'Transaction'  }  ],  fiscalYear: {  type: mongoose.Schema.Types.ObjectId,  ref: 'FiscalYear' // New field to reference the current fiscal year  },  defaultCashAccount: {  type: Boolean,  default: false // Flag to mark as default cash account  },  createdAt: {  type: Date,  default: Date.now() // Default creation timestamp  }  });  // Index to ensure unique account names within a company  accountSchema.index({ name: 1, company: 1, fiscalYear: 1 }, { unique: true });  module.exports = mongoose.model('Account', accountSchema); |

**1.3 Schema Fields**

* **name**: String, required; the name of the account.
* **address**: String, optional; the address of the account holder.
* **ward**: Number, optional; the ward number associated with the account.
* **phone**: String, optional; the contact number of the account holder.
* **pan**: Number, optional; the PAN number, must be at least 9 digits.
* **contactperson**: String, optional; name of the contact person.
* **email**: String, optional; email address of the account holder.
* **openingBalanceByFiscalYear**: Array of objects, containing opening balances for different fiscal years.
* **openingBalance**: Object; contains the opening balance amount and its type (Debit/Credit).
* **openingBalanceDate**: Date, defaulting to '2023-07-17'; the date for the opening balance.
* **companyGroups**: ObjectId, required; references the CompanyGroup model.
* **date**: Date, defaults to the current date; the creation date of the account.
* **company**: ObjectId, required; references the Company model.
* **transactions**: Array of ObjectIds, references to Transaction documents.
* **fiscalYear**: ObjectId; references the current fiscal year.
* **defaultCashAccount**: Boolean, default false; marks if this is the default cash account.
* **createdAt**: Date, defaults to the current date; timestamp of account creation.

**1.4 Validation**

* Ensures unique account names within the same company and fiscal year using an index.
* Required fields must be populated before saving the account to the database.
* **2. Company Routes**
* **2.1 Overview**
* The route handles the retrieval of companies based on specific conditions such as authentication, company selection, trade type, and fiscal year.

### 2.2 Route Definition

|  |
| --- |
| router.get('/companies', ensureAuthenticated, ensureCompanySelected, ensureTradeType, ensureFiscalYear, checkFiscalYearDateRange, async (req, res) => {  if (req.tradeType === 'Wholeseller') {  const companyId = req.session.currentCompany;  const currentCompanyName = req.session.currentCompanyName;  // Fetch the company and populate the fiscalYear  const company = await Company.findById(companyId).populate('fiscalYear');  // Check if fiscal year is already in the session or available in the company  let fiscalYear = req.session.currentFiscalYear ? req.session.currentFiscalYear.id : null;  let currentFiscalYear = null;  if (fiscalYear) {  currentFiscalYear = await FiscalYear.findById(fiscalYear);  }  if (!currentFiscalYear && company.fiscalYear) {  currentFiscalYear = company.fiscalYear;  // Set the fiscal year in the session for future requests  req.session.currentFiscalYear = {  id: currentFiscalYear.\_id.toString(),  startDate: currentFiscalYear.startDate,  endDate: currentFiscalYear.endDate,  name: currentFiscalYear.name,  dateFormat: currentFiscalYear.dateFormat,  isActive: currentFiscalYear.isActive  };  fiscalYear = req.session.currentFiscalYear.id;  }  if (!fiscalYear) {  return res.status(400).json({ error: 'No fiscal year found in session or company.' });  }  const accounts = await Account.find({  company: companyId,  fiscalYear: fiscalYear  }).populate('companyGroups');  const companyGroups = await CompanyGroup.find({ company: companyId });  res.render('wholeseller/company/companies', {  accounts,  companyGroups,  companyId,  currentCompanyName,  title: 'Account',  body: 'wholeseller >> account',  isAdminOrSupervisor: req.user.isAdmin || req.user.role === 'Supervisor'  });  }  }); |

**2.3 Route Flow**

1. **Authentication & Middleware Checks**: Ensures that the user is authenticated and has selected a company, trade type, and fiscal year.
2. **Fetch Company**: Retrieves the company from the database and populates its fiscal year.
3. **Fiscal Year Handling**:
   * Checks if a fiscal year is already stored in the session.
   * If not, retrieves it from the company object and saves it to the session for future requests.
4. **Account Retrieval**: Queries the Account collection for accounts associated with the selected company and fiscal year.
5. **Render Response**: Renders the companies view, passing in the accounts, company groups, and relevant metadata.

**2.4 Error Handling**

* Returns a 400 status code with an error message if no fiscal year is found in the session or associated with the company.

**2.5 Expected Outputs**

* On success, renders a view displaying the accounts and their associated company groups for a wholesaler.
* On failure, responds with an appropriate error message in JSON format.

**3. Testing Guidelines**

**3.1 Test Cases for Account Schema**

* Verify that an account can be created with valid fields.
* Ensure that duplicate account names within the same company and fiscal year cannot be created (validation).
* Check that all required fields are enforced.

**3.2 Test Cases for Company Routes**

* Test successful retrieval of companies for a wholesaler with valid session data.
* Verify that unauthorized users receive an appropriate error message.
* Ensure that the application handles cases where no fiscal year is available.

**4.0 POST Route for Creating Companies** (routes/companies.js)

This route handles the creation of new company accounts with validation for unique names and correct association with fiscal years and company groups.

**Route Flow:**

1. **Route Protection**:
   * Similar middleware as the GET route is applied to ensure authenticated requests and the proper setup of fiscal year and company data.

**Request Data**:

* Retrieves the account details (e.g., name, address, phone, ward, pan, email, contact person, opening balance, company groups) from req.body.

**Company and Fiscal Year Handling**:

* Company ID and fiscal year are checked and fetched from the session or database, ensuring they are valid.

**Validation**:

* Validates the existence of the selected company group (CompanyGroup).
* Checks for duplicate accounts using the unique index on the name, company, and fiscalYear.

**Account Creation**:

* A new Account document is created with the provided details and saved to the database.

**Error Handling**:

* Duplicate account names trigger a flash message.
* Server errors result in a 500 status code and an appropriate error message.

**Redirection**:

* Upon success, the user is redirected to the /companies page with a success message.

**Testing Focus:**

* Ensure that the form validation works for all required fields (e.g., name, company group).
* Test for the uniqueness constraint on account names.
* Validate proper association of the account with the company and fiscal year.
* Check error handling, especially for duplicate account names or missing fiscal year.
* Verify that the correct flash messages are shown on success or error.
* Test the redirection flow after successfully creating a new account.

**4. Middleware Usage**

Several middlewares are used to protect and validate the routes:

* **ensureAuthenticated**: Ensures the user is logged in.
* **ensureCompanySelected**: Validates that a company is selected before proceeding.
* **ensureTradeType**: Checks that the company trade type matches (e.g., Wholeseller).
* **ensureFiscalYear**: Ensures the fiscal year is set before proceeding.
* **checkFiscalYearDateRange**: Validates that the fiscal year is within the proper date range.

**Testing Focus:**

* Test each middleware to ensure proper protection and validation of the routes.
* Ensure unauthorized access is correctly blocked and error messages are displayed.
* Validate that session data (e.g., company, fiscal year) is correctly set and retrieved.

**5. Additional Considerations**

**Database Indexing:**

* The unique index on the accountSchema for name, company, and fiscalYear needs to be tested for performance and correctness during large data sets.

**Date and Time Handling:**

* Verify that dates, such as openingBalanceDate and createdAt, are handled correctly, especially with timezone differences.

**Route: View Company Account:**

**URL**: /companies/:id

**Method**: GET

**Middleware**: ensureAuthenticated, ensureCompanySelected, ensureTradeType, ensureFiscalYear, checkFiscalYearDateRange

**Description**: Displays the details of a specific company account for the authenticated user.

**Functionality:**

* Fetches the account details based on the account ID and current fiscal year.
* Validates if the account belongs to the current company using the company.\_id field.
* Renders the view page to display the company account details, company groups, and current company name.

**Error Handling:**

* If the account does not belong to the current company, returns a 403 Unauthorized error.
* Logs and displays errors for issues encountered while fetching the company details.

**3. Route: Edit Company Account**

* **URL**: /companies/:id/edit
* **Method**: GET
* **Middleware**: ensureAuthenticated, ensureCompanySelected, ensureTradeType, ensureFiscalYear, checkFiscalYearDateRange
* **Description**: Renders a form to edit the details of an existing company account.

**Functionality:**

* Fetches the account details using the account ID and checks if it belongs to the current company.
* Populates the form with the account and company group details for the authenticated user to edit.

**Error Handling:**

* If the account does not belong to the current company, returns a 403 Unauthorized error.
* Logs and displays errors for issues encountered while fetching the company details.

**4. Route: Update Company Account**

* **URL**: /companies/:id
* **Method**: PUT
* **Middleware**: ensureAuthenticated, ensureCompanySelected, ensureTradeType, ensureFiscalYear, checkFiscalYearDateRange
* **Description**: Updates the details of a specific company account with new information.

**Request Parameters:**

* **Body**:
  + Similar to the create route: name, address, phone, pan, contactperson, email, companyGroups, and openingBalance.

**Functionality:**

* Fetches the current fiscal year and validates the company group.
* Updates the account information with the new data.
* Ensures the openingBalance is associated with the fiscal year.
* Redirects to /companies upon successful update.

**Error Handling:**

* If a duplicate account name exists within the selected company, it returns a duplicate key error.
* Logs and displays errors for issues encountered while updating the company.

**5. Route: Delete Company Account**

* **URL**: /companies/:id
* **Method**: DELETE
* **Middleware**: ensureAuthenticated, ensureCompanySelected, ensureTradeType, ensureFiscalYear, checkFiscalYearDateRange
* **Description**: Deletes a company account if it is not the default cash account and has no associated transactions.

**Functionality:**

* Verifies if the account is a default cash account and prevents deletion if true.
* Checks if there are any associated transactions with the account. If transactions exist, deletion is prevented.
* Deletes the account if the conditions are met and redirects to /companies.

**Error Handling:**

* If the account is a default cash account or has associated transactions, returns a 400 Bad Request.
* Logs and displays errors for issues encountered while deleting the account.