{

"name": "Gaurav Raj",

"tech": "GATE",

"sub" : "true"

}

|  |
| --- |
| 1. const mongoose = require('mongoose')  * **Purpose**: This line imports the mongoose library, which is used to interact with MongoDB, a NoSQL database. * **Explanation**: mongoose provides a straightforward way to model data, create schemas, and perform database operations in Node.js.  2. const userSchema = new mongoose.Schema({  * **Purpose**: This creates a new schema for a User model. * **Explanation**:   + A schema defines the structure of the documents in a MongoDB collection.   + It specifies what fields the documents will have, their data types, and any validation rules.  3. name: {  * **Purpose**: This defines the name field in the schema. * **Explanation**:   + type: String ensures that the value of name must be a string.   + required: true makes this field mandatory. If not provided, an error will occur when trying to save a document.  4. tech: {  * **Purpose**: This defines the tech field in the schema. * **Explanation**:   + type: String indicates that this field must also be a string.   + required: true means that the tech field must be provided when creating or updating a document.  5. sub: {  * **Purpose**: This defines the sub field in the schema. * **Explanation**:   + type: Boolean ensures that the value is either true or false.   + required: true makes this field mandatory.   + default: false sets the default value of sub to false if no value is explicitly provided.  6. })  * **Purpose**: This closes the schema definition. * **Explanation**: All fields and their respective configurations are enclosed within this object.  7. module.exports = mongoose.model('User', userSchema)  * **Purpose**: This creates a model named User based on the userSchema and exports it. * **Explanation**:   + mongoose.model('User', userSchema):     - Creates a collection called users in the MongoDB database (Mongoose automatically pluralizes the model name).     - Maps the userSchema to this collection, enabling CRUD (Create, Read, Update, Delete) operations.   + module.exports allows this model to be used in other files by importing it using require. |
| Summary of Functionality: This code:   1. Defines a User schema with fields name, tech, and sub. 2. Ensures proper data validation and default values. 3. Exports a User model to interact with the users collection in MongoDB.   You can now use this model to perform database operations like adding a new user, retrieving users, updating records, or deleting records. |
| The string 'User' passed as the first argument to mongoose.model determines the **name of the collection** in MongoDB. Mongoose **pluralizes** and **lowercases** this name automatically to decide the collection name. Behavior for 'User' module.exports = mongoose.model('User', userSchema);   * The model name is 'User'. * Mongoose creates or maps this to a MongoDB collection named users (lowercase and pluralized).  Behavior for 'user' module.exports = mongoose.model('user', userSchema); |
| * The model name is 'user'. * Mongoose creates or maps this to a MongoDB collection named users (still lowercase and pluralized).  Key Point: The result is the **same** in this specific case because:   1. 'User' is case-insensitively converted to lowercase. 2. 'user' is already lowercase. 3. Both names are pluralized to users.  When It Matters: If you use names like 'USER', 'UserDetail', or 'user\_detail', Mongoose will process them differently:   * 'USER' → users * 'UserDetail' → userdetails * 'user\_detail' → user\_details  Conclusion: For 'User' and 'user', there's no difference in behavior because both resolve to the same users collection in MongoDB. However, it’s good practice to use consistent and descriptive naming conventions, usually starting with an uppercase letter for model names ('User'). |
|  |