**Scenario**

Let us say you are creating a To-do application and you want the goals (to-dos) to be associated with a particular user. You would create a schema for Users and a schema for goals. **The question is,** How do you attach the goals to a user.

**Ref option**

Now, you can reference documents in other collections. You can replace a specified path in a document with document(s) from other collection(s), this process is known as **population**

The ref option is what tells mongoose.js which model to use during population.

**Example**

userModel.js

const userSchema = schema({

user: {

type: String

required: [true, "add a name"]

},

})

goalModel.js

const goalSchema = schema({

user: {

type: mongoose.Schema.Types.ObjectId,

ref: "User",

required: true

},

text: {

type: String

}

})

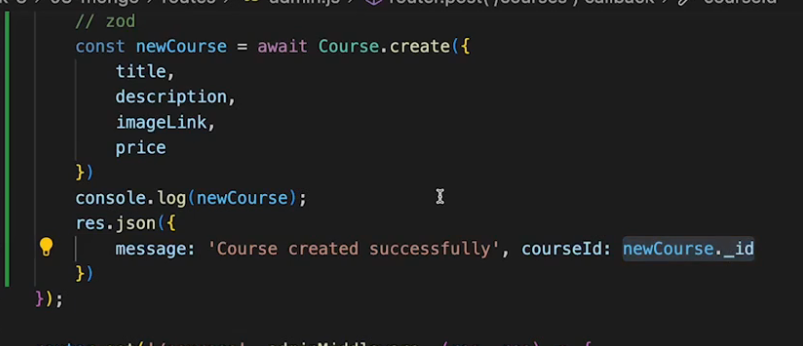
**Explaination**

In our goalModel, it contains a user field which is set to type ObjetId, and with the ref option, we have told mongoose to use the id from our userModel to fill the user field in goalModel during population.

This means all \_id we store in the user field of goalModel must be document \_id from the User Model.

[Models](https://mongoosejs.com/docs/api/model.html) are fancy constructors compiled from Schema definitions. An instance of a model is called a [document](https://mongoosejs.com/docs/documents.html). Models are responsible for creating and reading documents from the underlying MongoDB database.

Create method returns a newly created document.



https://coursework.vschool.io/mongoose-crud/