

Node Js Ödevi

13th Week

Farhan Ahmad

20360859096

Geçen hafta **MongoDB ve Promiseler** gibi konuları incelemiştik. Bu hafta ise gibi **Mongoose ve validator Kütüphanesinin Kurulumu ve Kullanımı** konuları ele alacağız.

Bu raporda, MongoDB veritabanı ile birlikte Mongoose kullanarak nasıl veri modeli oluşturulacağı ve nasıl veri eklenip doğrulama yapılacağı adım adım anlatılacaktır.

Mongoose Kurulumu:

Mongoose, MongoDB ile çalışırken kullanılabilen bir Object-Document Mapper (ODM) aracıdır. Mongoose, doğrulama, isteğe bağlı/zorunlu alanlar gibi işlemleri yönetir.

Gerekli Kurulumlar:

npm i mongoose

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Farhan Ahmad\Desktop\node\week13> npm i mongoose
added 8 packages, changed 1 package, and audited 137 packages in 10s
15 packages are looking for funding
  run `npm fund` for details
2 moderate severity vulnerabilities
Some issues need review, and may require choosing
a different dependency.
```

Mongoose Bağlantısı ve Model Oluşturma:

Yeni bir Mongoose modeli oluşturmak için aşağıdaki adımları takip edin:

```
const mongoose = require('mongoose');

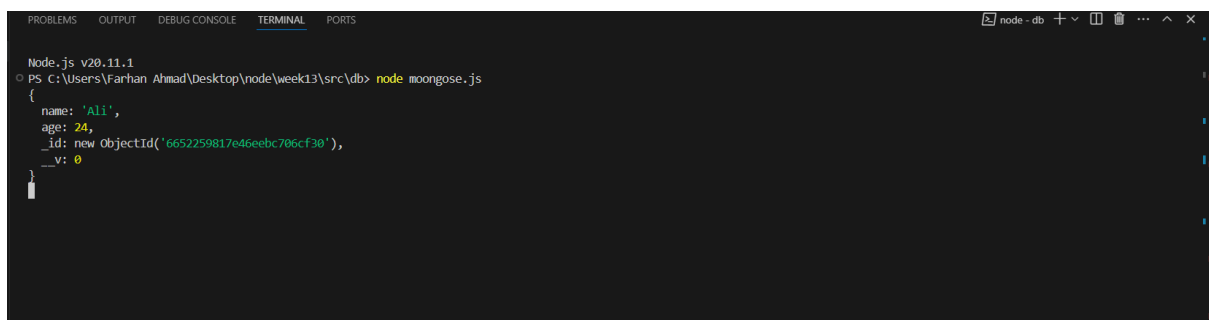
mongoose.connect('mongodb://127.0.0.1:27017/task-manager-api', {
});

const User = mongoose.model('User', {
  name: {
    type: String
  },
  age: {
    type: Number
  }
});

const me = new User({
  name: 'Ali',
  age: 24
});

me.save().then(() => {
  console.log(me);
}).catch((error) => {
  console.log('Error!', error);
});
```

Bu kod parçasında, Task adında yeni bir model oluşturulmuş ve bu model ile yeni bir görev eklenmiştir ve sonuç aşağıdaki gibi alabiliriz.



```
Node.js v20.11.1
PS C:\Users\Farhan Ahmad\Desktop\node\week13\src\db> node mongoose.js
{
  name: 'Ali',
  age: 24,
  _id: new ObjectId('6652259817e46eebc706cf30'),
  __v: 0
}
```

Veri Doğrulama ve Temizleme:

Veri doğrulama ve temizleme işlemleri aşağıdaki gibi yapılabilir:

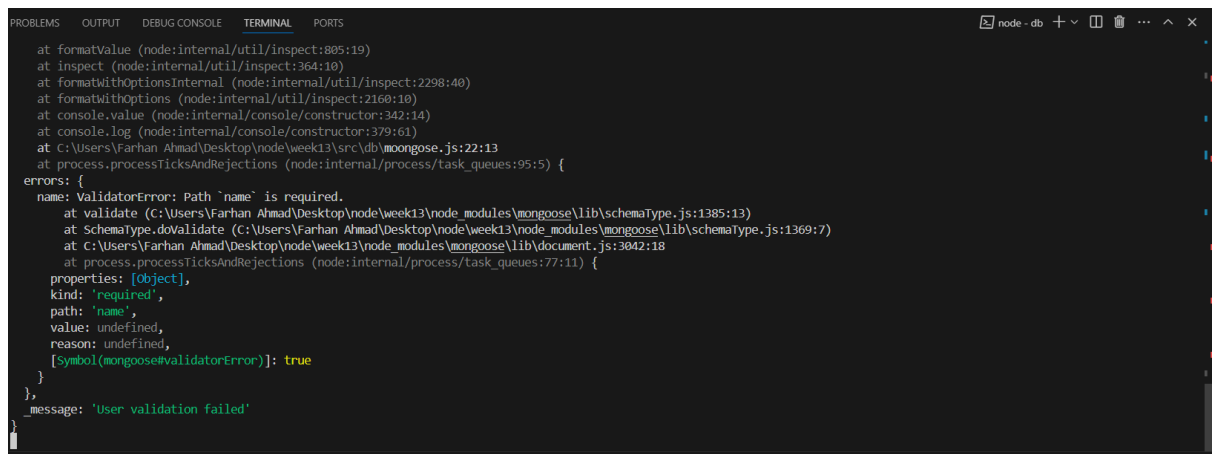
- User modelini güncellemek:

```
const User = mongoose.model('User', {
  name: {
    type: String,
    required: true
  },
  age: {
    type: Number
  }
});

const me = new User({
});

me.save().then(() => {
  console.log(me);
}).catch((error) => {
  console.log('Error!', error);
});
```

Bu kod parçasında, name alanının zorunlu olduğu belirtilmiştir. name alanı boş olduğunda hata mesajı dönecektir.



```
node - db + v [ ] [ ] ... ^ x
at formatValue (node:internal/util/inspect:805:19)
at inspect (node:internal/util/inspect:364:10)
at formatWithOptionsInternal (node:internal/util/inspect:2298:40)
at formatWithOptions (node:internal/util/inspect:2160:10)
at console.value (node:internal/console/constructor:342:14)
at console.log (node:internal/console/constructor:379:61)
at C:\Users\Farhan Ahmad\Desktop\node\week13\src\db\mongoose.js:22:13
at process.processTicksAndRejections (node:internal/process/task_queues:95:5) {
  errors: {
    name: ValidatorError: Path `name` is required.
      at validate (C:\Users\Farhan Ahmad\Desktop\node\week13\node_modules\mongoose\lib\schematype.js:1385:13)
      at SchemaType.doValidate (C:\Users\Farhan Ahmad\Desktop\node\week13\node_modules\mongoose\lib\schematype.js:1369:7)
      at C:\Users\Farhan Ahmad\Desktop\node\week13\node_modules\mongoose\lib\document.js:3042:18
      at process.processTicksAndRejections (node:internal/process/task_queues:77:11) {
        properties: [Object],
        kind: 'required',
        path: 'name',
        value: undefined,
        reason: undefined,
        [Symbol(mongoose#validatorError)]: true
      }
    },
    message: 'User validation failed'
  }
```

Bu hatayı gidermek için aşağıdaki kodları takip edelim.

İsim Bilgisini Sağlama:

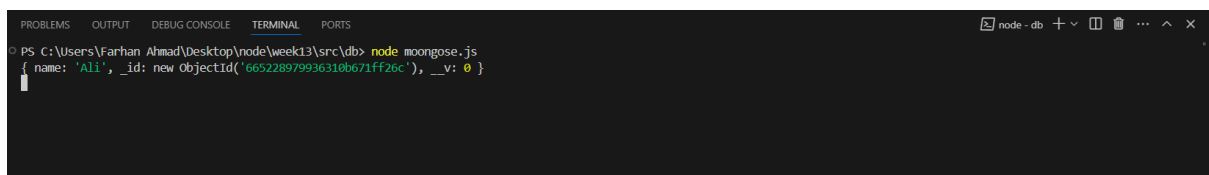
mongoose.js dosyasını aşağıdaki gibi güncelleyin:

```
const User = mongoose.model('User', {
  name: {
    type: String
  },
  age: {
    type: Number
  }
});

const me = new User({
  name: 'Ali'
});

me.save().then(() => {
  console.log(me);
}).catch((error) => {
  console.log('Error!', error);
});
```

Bu kod parçasında, yalnızca name alanı sağlanmış ve kullanıcı kaydedilmiştir.



```
node - db
PS C:\Users\Farhan Ahmad\Desktop\node\week13\src\db> node mongoose.js
{ name: 'Ali', _id: new ObjectId('665228979936310b671ff26c'), __v: 0 }
```

Özel Doğrulayıcı Tanımlama:

Yaşın pozitif bir değer olmasını sağlamak için özel bir doğrulayıcı ekleyelim:

```
const User = mongoose.model('User', {
  name: {
    type: String,
    required: true
  },
  age: {
```

```

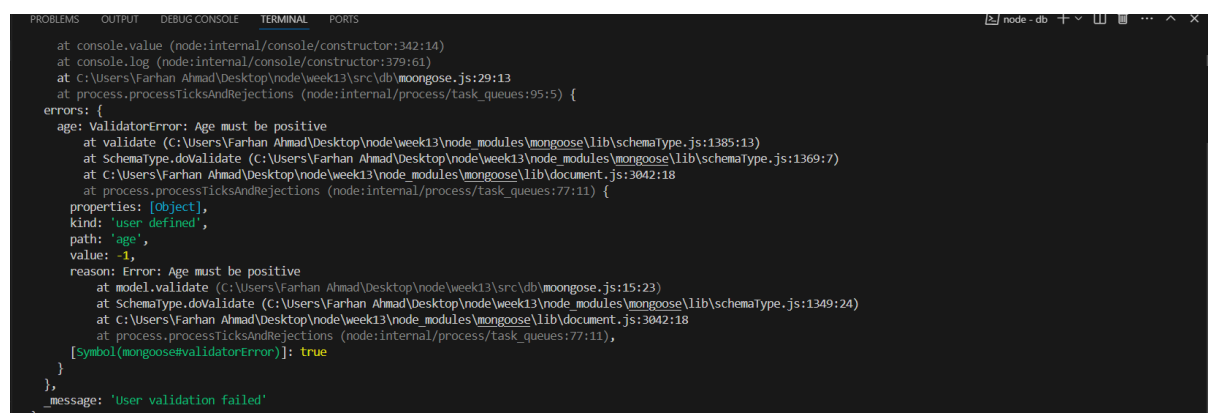
        type: Number,
        validate(value) {
            if (value < 0) {
                throw new Error('Age must be positive');
            }
        }
    }
});

const me = new User({
    name: 'Ali',
    age: -1
});

me.save().then(() => {
    console.log(me);
}).catch((error) => {
    console.log('Error!', error);
});

```

Bu kod parçasında, negatif bir yaş değeri sağlandığı için aşağıdaki gibi hata mesajı dönecektir.



```

node - db
at console.value (node:internal/console/constructor:342:14)
at console.log (node:internal/console/constructor:379:61)
at C:\Users\Farhan Ahmad\Desktop\node\week13\src\db\mongoose.js:29:13
at process.processTicksAndRejections (node:internal/process/task_queues:95:5) {
  errors: {
    age: ValidatorError: Age must be positive
      at validate (C:\Users\Farhan Ahmad\Desktop\node\week13\node_modules\mongoose\lib\schematype.js:1385:13)
      at SchemaType.doValidate (C:\Users\Farhan Ahmad\Desktop\node\week13\node_modules\mongoose\lib\schematype.js:1369:7)
      at C:\Users\Farhan Ahmad\Desktop\node\week13\node_modules\mongoose\lib\document.js:3042:18
      at process.processTicksAndRejections (node:internal/process/task_queues:77:11) {
        properties: [Object],
        kind: 'user_defined',
        path: 'age',
        value: -1,
        reason: Error: Age must be positive
          at model.validate (C:\Users\Farhan Ahmad\Desktop\node\week13\src\db\mongoose.js:15:23)
          at SchemaType.doValidate (C:\Users\Farhan Ahmad\Desktop\node\week13\node_modules\mongoose\lib\schematype.js:1349:24)
          at C:\Users\Farhan Ahmad\Desktop\node\week13\node_modules\mongoose\lib\document.js:3042:18
          at process.processTicksAndRejections (node:internal/process/task_queues:77:11),
        [Symbol(mongoose#validatorError)]: true
      }
    },
    _message: 'User validation failed'
  }
}

```

validator.js Kütüphanesini Kullanma:

E-posta gibi verileri doğrulamak için validator.js kütüphanesini kullanın:

```

const mongoose = require('mongoose');
const validator = require('validator');
mongoose.connect('mongodb://127.0.0.1:27017/task-manager-api', {

```

```

});
const User = mongoose.model('User', {
  name: {
    type: String,
    required: true
  },
  email: {
    type: String,
    required: true,
    validate(value) {
      if (!validator.isEmail(value)) {
        throw new Error('Email is invalid');
      }
    }
  },
  age: {
    type: Number,
    validate(value) {
      if (value < 0) {
        throw new Error('Age must be positive');
      }
    }
  }
});

const me = new User({
  name: 'Ali',
  email: 'ali@'
});

me.save().then(() => {
  console.log(me);
}).catch((error) => {
  console.log('Error!', error);
});

```

Bu kod parçasında, geçersiz bir e-posta sağlandığında aşağıdaki gibi hata mesajı dönecektir.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
node - db + - [ ] ... ^ x

at console.value (node:internal/console/constructor:342:14)
at console.log (node:internal/console/constructor:379:61)
at C:\Users\Farhan Ahmad\Desktop\node\week13\src\db\mongoose.js:38:13
at process.processTicksAndRejections (node:internal/process/task_queues:95:9) {
  errors: {
    email: ValidatorError: Email is invalid
      at validate (C:\Users\Farhan Ahmad\Desktop\node\week13\node_modules\mongoose\lib\schemaType.js:1385:13)
      at SchemaType.doValidate (C:\Users\Farhan Ahmad\Desktop\node\week13\node_modules\mongoose\lib\schemaType.js:1369:7)
      at C:\Users\Farhan Ahmad\Desktop\node\week13\node_modules\mongoose\lib\document.js:3042:18
      at process.processTicksAndRejections (node:internal/process/task_queues:77:11) {
        properties: [Object],
        kind: 'user defined',
        path: 'email',
        value: 'ali@',
        reason: Error: Email is invalid
          at model.validate (C:\Users\Farhan Ahmad\Desktop\node\week13\src\db\mongoose.js:16:23)
          at SchemaType.doValidate (C:\Users\Farhan Ahmad\Desktop\node\week13\node_modules\mongoose\lib\schemaType.js:1349:24)
          at C:\Users\Farhan Ahmad\Desktop\node\week13\node_modules\mongoose\lib\document.js:3042:18
          at process.processTicksAndRejections (node:internal/process/task_queues:77:11),
        [Symbol(mongoose#validatorError)]: true
      }
    },
    _message: 'User validation failed'
  }
}
```

Veriler Üzerinde Daha Fazla Kontrol:

Veri temizleme ve ek doğrulamalar ekleyelim:

```
const User = mongoose.model('User', {
  name: {
    type: String,
    required: true,
    trim: true
  },
  email: {
    type: String,
    required: true,
    trim: true,
    lowercase: true,
    validate(value) {
      if (!validator.isEmail(value)) {
        throw new Error('Email is invalid');
      }
    }
  },
  age: {
    type: Number,
    default: 0,
    validate(value) {
      if (value < 0) {
        throw new Error('Age must be positive');
      }
    }
  }
})
```

```
});

const me = new User({
  name: ' Ali. ',
  email: 'ali@aVVVVVc.com. '
});

me.save().then(() => {
  console.log(me);
}).catch((error) => {
  console.log('Error!', error);
});
```

Bu kod parçasında, name ve email alanları trimlenmiş, e-posta küçük harfe çevrilmiş ve geçersiz e-posta durumunda aşağıdaki gibi hata mesajı dönecektir.

```
node - db
at console.value (node:internal/console/constructor:342:14)
at console.log (node:internal/console/constructor:379:61)
at C:\Users\Farhan Ahmad\Desktop\node\week13\src\db\mongoose.js:42:13
at process.processTicksAndRejections (node:internal/process/task_queues:95:5) {
  errors: {
    email: ValidatorError: Email is invalid
      at validate (C:\Users\Farhan Ahmad\Desktop\node\week13\node_modules\mongoose\lib\schemaType.js:1385:13)
      at SchemaType.doValidate (C:\Users\Farhan Ahmad\Desktop\node\week13\node_modules\mongoose\lib\schemaType.js:1369:7)
      at C:\Users\Farhan Ahmad\Desktop\node\week13\node_modules\mongoose\lib\document.js:3042:18
      at process.processTicksAndRejections (node:internal/process/task_queues:77:11) {
        properties: [Object],
        kind: 'user defined',
        path: 'email',
        value: 'ali@aVVVVVc.com.',
        reason: Error: Email is invalid
          at model.validate (C:\Users\Farhan Ahmad\Desktop\node\week13\src\db\mongoose.js:19:23)
          at SchemaType.doValidate (C:\Users\Farhan Ahmad\Desktop\node\week13\node_modules\mongoose\lib\schemaType.js:1349:24)
          at C:\Users\Farhan Ahmad\Desktop\node\week13\node_modules\mongoose\lib\document.js:3042:18
          at process.processTicksAndRejections (node:internal/process/task_queues:77:11),
        [Symbol(mongoose#validatorError)]: true
      }
    },
    _message: 'User validation failed'
  }
}
```

Kullanıcı Modeline Şifre Alanı Ekleme:

Kullanıcı modeline şifre alanı ekleyerek bu alanın zorunlu, en az 7 karakter uzunluğunda, trimlenmiş olmasını ve "password" kelimesini içermemesini sağlayacağız.

Kullanıcı Modelini Güncelleme

mongoose.js dosyasını aşağıdaki gibi güncelleyelim:


```

const User = mongoose.model('User', {
  name: {
    type: String,
    required: true,
    trim: true
  },
  email: {
    type: String,
    required: true,
    trim: true,
    lowercase: true,
    validate(value) {
      if (!validator.isEmail(value)) {
        throw new Error('Email is invalid');
      }
    }
  },
  password: {
    type: String,
    required: true,
    minlength: 7,
    trim: true,
    validate(value) {
      if (value.toLowerCase().includes('password')) {
        throw new Error('Password cannot contain "password"');
      }
    }
  },
  age: {
    type: Number,
    default: 0,
    validate(value) {
      if (value < 0) {
        throw new Error('Age must be positive');
      }
    }
  }
});

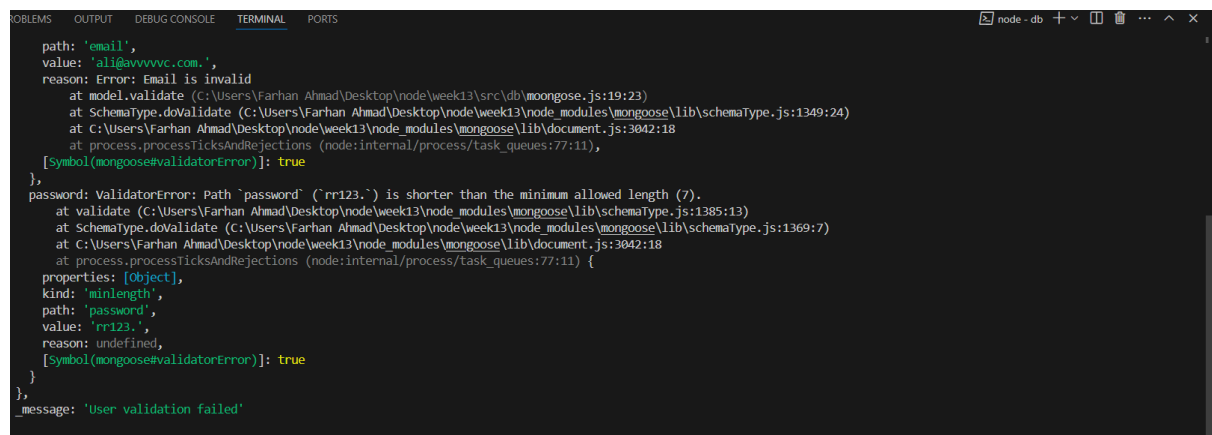
const me = new User({
  name: 'Ali.',
  email: 'ali@aVVVVc.com.',
  password: 'rr123.'
});

me.save().then(() => {

```

```
    console.log(me);
  }).catch((error) => {
    console.log('Error!', error);
  });
```

Bu kod parçasında, password alanı eklenmiş ve gerekli doğrulamalar yapılmıştır. Ve geçersiz password durumunda aşağıdaki gibi hata mesajı dönecektir.



```
path: 'email',
value: 'ali@avvvvc.com.',
reason: Error: Email is invalid
    at model.validate (C:\Users\Farhan Ahmad\Desktop\node\week13\src\db\mongoose.js:19:23)
    at SchemaType.doValidate (C:\Users\Farhan Ahmad\Desktop\node\week13\node_modules\mongoose\lib\schematype.js:1349:24)
    at C:\Users\Farhan Ahmad\Desktop\node\week13\node_modules\mongoose\lib\document.js:3042:18
    at process.processTicksAndRejections (node:internal/process/task_queues:77:11),
[Symbol(mongoose#validatorError)]: true
},
password: ValidatorError: Path 'password' ('rr123.') is shorter than the minimum allowed length (7).
    at validate (C:\Users\Farhan Ahmad\Desktop\node\week13\node_modules\mongoose\lib\schematype.js:1385:13)
    at SchemaType.doValidate (C:\Users\Farhan Ahmad\Desktop\node\week13\node_modules\mongoose\lib\schematype.js:1369:7)
    at C:\Users\Farhan Ahmad\Desktop\node\week13\node_modules\mongoose\lib\document.js:3042:18
    at process.processTicksAndRejections (node:internal/process/task_queues:77:11) {
  properties: [Object],
  kind: 'minlength',
  path: 'password',
  value: 'rr123.',
  reason: undefined,
  [Symbol(mongoose#validatorError)]: true
},
},
message: 'User validation failed'
```

Görev Modelini Özelleştirme:

Görev modeline doğrulama ekleyerek açıklama alanını zorunlu ve trimlenmiş yapacağız. Tamamlanma alanını ise isteğe bağlı yapıp varsayılan olarak false ayarlayacağız.

Görev Modelini Güncelleme:

mongoose.js dosyasını aşağıdaki gibi güncelleyelim:

```
const Task = mongoose.model('Task', {
  description: {
    type: String,
    required: true,
    trim: true
  },
  completed: {
    type: Boolean,
```

```

        default: false
      }
    });

const task1 = new Task({
});

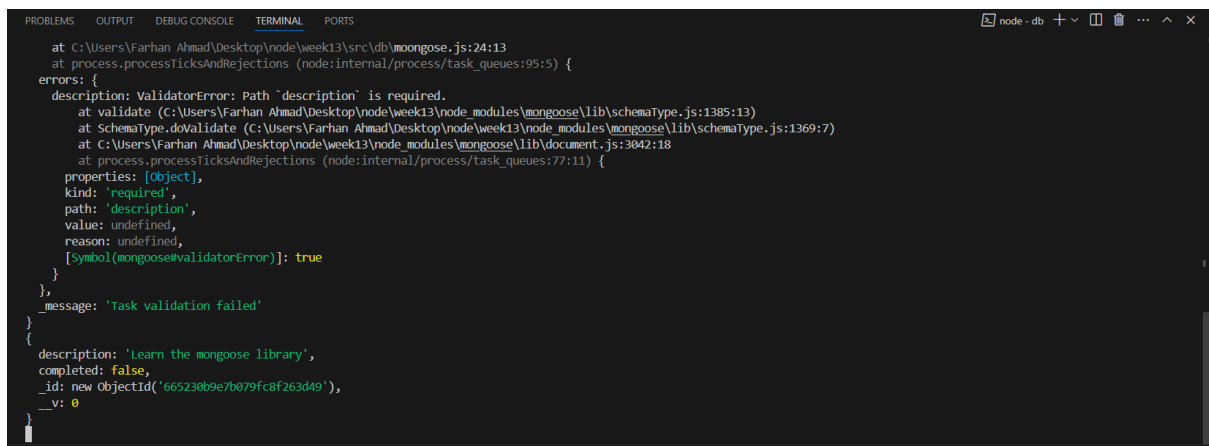
task1.save().then(() => {
  console.log(task1);
}).catch((error) => {
  console.log('Error!', error);
});

const task2 = new Task({
  description: 'Learn the mongoose library'
});

task2.save().then(() => {
  console.log(task2);
}).catch((error) => {
  console.log('Error!', error);
});

```

Bu kod parçasında, description alanı zorunlu ve trimlenmiş olarak ayarlanmış, completed alanı ise isteğe bağlı olarak varsayılan false değeri ile ayarlanmıştır.



```

node - db
at C:\Users\Farhan Ahmad\Desktop\node\week13\src\db\mongoose.js:24:13
at process.processTicksAndRejections (node:internal/process/task_queues:95:5) {
  errors: {
    description: ValidatorError: Path "description" is required.
      at validate (C:\Users\Farhan Ahmad\Desktop\node\week13\node_modules\mongoose\lib\schematype.js:1385:13)
      at SchemaType.doValidate (C:\Users\Farhan Ahmad\Desktop\node\week13\node_modules\mongoose\lib\schematype.js:1369:7)
      at C:\Users\Farhan Ahmad\Desktop\node\week13\node_modules\mongoose\lib\document.js:3042:18
      at process.processTicksAndRejections (node:internal/process/task_queues:77:11) {
        properties: [Object],
        kind: 'required',
        path: 'description',
        value: undefined,
        reason: undefined,
        [Symbol(mongoose#validatorError)]: true
      }
    },
    _message: 'Task validation failed'
  }
  {
    description: 'Learn the mongoose library',
    completed: false,
    _id: new ObjectId('665230b9e7b079fc8f263d49'),
    __v: 0
  }
}

```

Bu adımlarla birlikte, Mongoose kullanarak kullanıcı ve görev modelleri için gelişmiş doğrulama ve veri temizleme işlemlerini başarıyla yapabiliriz. Dosyacımızı kaydedip çalıştırarak sonuçları görebiliriz.