# Full Stack Development with MERN

# API Development and Integration Report

| | |
|---|---|
| Date | 16 July 2024 |
| Team ID | SWTID1719923176 |
| Project Name | Freelance-Finder |
| Maximum Marks | |

**Project Title:** Freelance Finder

**Date:** 16 July 2024
**Prepared by:** Aryaman Parashar Behera, Disha M, P R Adithya, Swarna Kamalam S

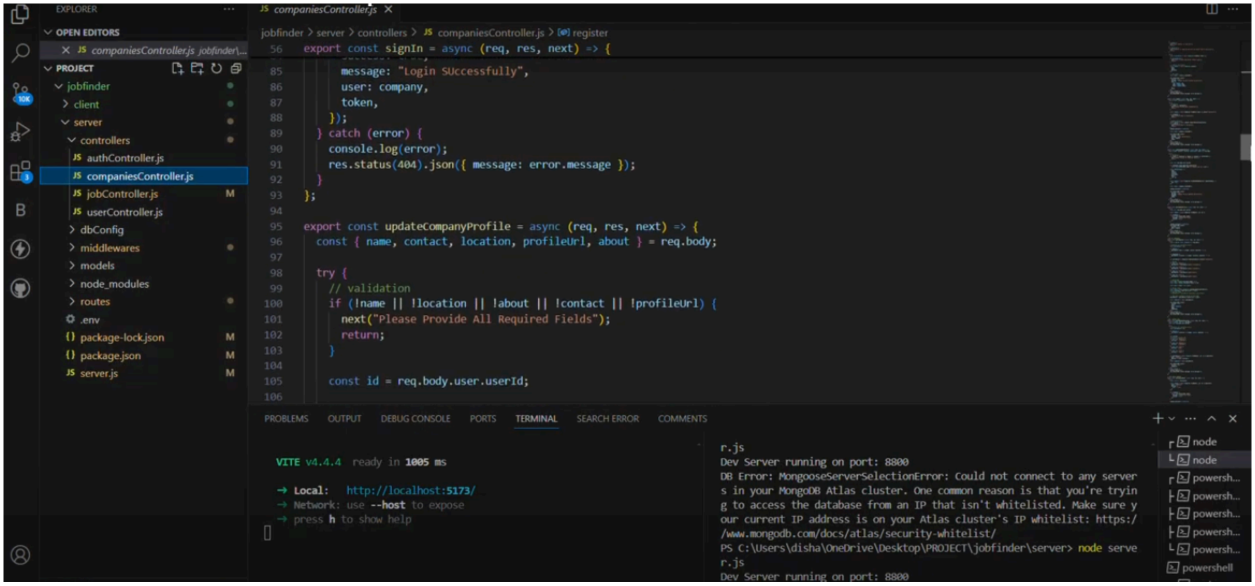
## Objective

The objective of this report is to document the API development progress and key aspects of the backend services implementation for the [Your Project Title] project.

## Technologies Used

- **Backend Framework:** Node.js with Express.js
- **Database:** MongoDB
- **Authentication:** JWT

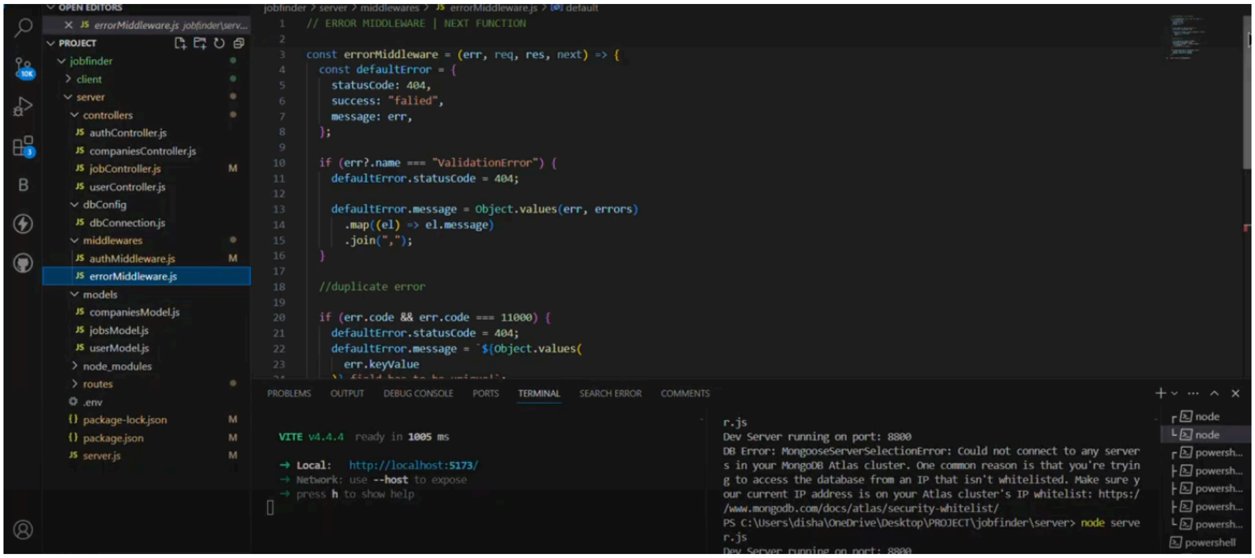
## Project Structure



```
export const signIn = async (req, res, next) => {
      message: "Login SUccessfully",
      user: company,
      token,
    });
  } catch (error) {
    console.log(error);
    res.status(404).json({ message: error.message });
  }
};

export const updateCompanyProfile = async (req, res, next) => {
  const { name, contact, location, profileUrl, about } = req.body;

  try {
    // validation
    if (!name || !location || !about || !contact || !profileUrl) {
      next("Please Provide All Required Fields");
      return;
    }

    const id = req.body.user.userId;
```

```
  VITE v4.4.4  ready in 1005 ms

  → Local:   http://localhost:5173/
  → Network: use --host to expose
  → press h to show help
```

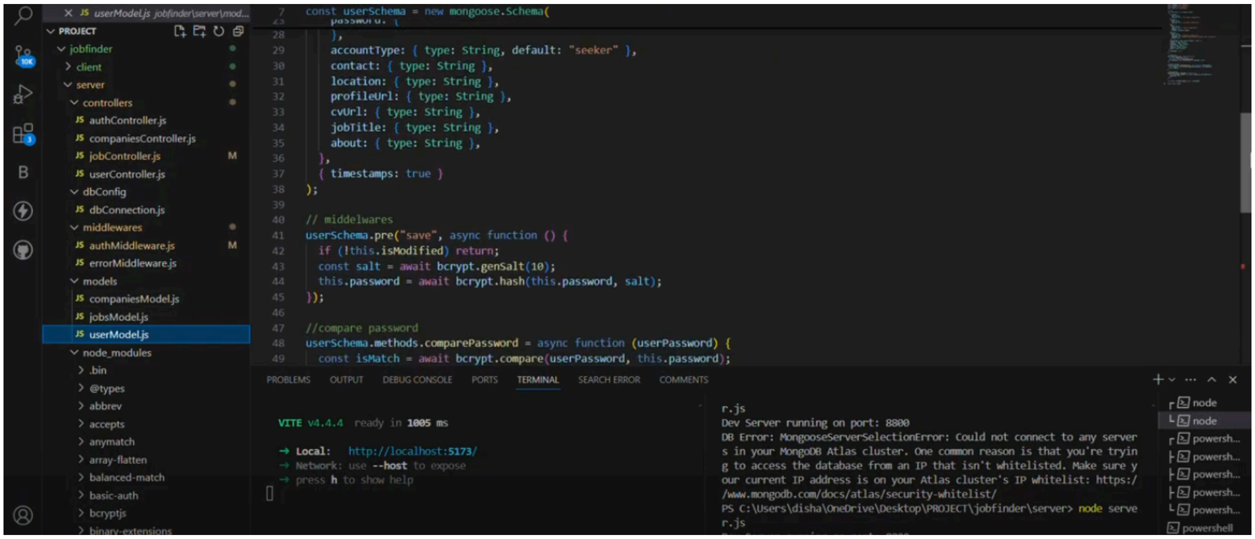
```
r.js
Dev Server running on port: 8800
DB Error: MongooseServerSelectionError: Could not connect to any server
s in your MongoDB Atlas cluster. One common reason is that you're tryin
g to access the database from an IP that isn't whitelisted. Make sure y
our current IP address is on your Atlas cluster's IP whitelist: https:/
/www.mongodb.com/docs/atlas/security-whitelist/
PS C:\Users\disha\OneDrive\Desktop\PROJECT\jobfinder\server> node serve
r.js
Dev Server running on port: 8800
```



```
// ERROR MIDDLEWARE | NEXT FUNCTION

const errorMiddleware = (err, req, res, next) => {
  const defaultError = {
    statusCode: 404,
    success: "falied",
    message: err,
  };

  if (err?.name === "ValidationError") {
    defaultError.statusCode = 404;

    defaultError.message = Object.values(err, errors)
      .map((el) => el.message)
      .join(",");
  }

  //duplicate error

  if (err.code && err.code === 11000) {
    defaultError.statusCode = 404;
    defaultError.message = `${Object.values(
      err.keyValue
```

```
VITE v4.4.4  ready in 1005 ms

→ Local:   http://localhost:5173/
→ Network: use --host to expose
→ press h to show help
```

```
r.js
Dev Server running on port: 8800
DB Error: MongooseServerSelectionError: Could not connect to any server
s in your MongoDB Atlas cluster. One common reason is that you're tryin
g to access the database from an IP that isn't whitelisted. Make sure y
our current IP address is on your Atlas cluster's IP whitelist: https:/
/www.mongodb.com/docs/atlas/security-whitelist/
PS C:\Users\disha\OneDrive\Desktop\PROJECT\jobfinder\server> node serve
r.js
Dev Server running on port: 8800
```



```
const userSchema = new mongoose.Schema(
    },
    accountType: { type: String, default: "seeker" },
    contact: { type: String },
    location: { type: String },
    profileUrl: { type: String },
    cvUrl: { type: String },
    jobTitle: { type: String },
    about: { type: String },
  },
  { timestamps: true }
);

// middelwares
userSchema.pre("save", async function () {
  if (!this.isModified) return;
  const salt = await bcrypt.genSalt(10);
  this.password = await bcrypt.hash(this.password, salt);
});

//compare password
userSchema.methods.comparePassword = async function (userPassword) {
  const isMatch = await bcrypt.compare(userPassword, this.password);
```

```
VITE v4.4.4  ready in 1005 ms

→ Local:   http://localhost:5173/
→ Network: use --host to expose
→ press h to show help
```

```
r.js
Dev Server running on port: 8800
DB Error: MongooseServerSelectionError: Could not connect to any server
s in your MongoDB Atlas cluster. One common reason is that you're tryin
g to access the database from an IP that isn't whitelisted. Make sure y
our current IP address is on your Atlas cluster's IP whitelist: https:/
/www.mongodb.com/docs/atlas/security-whitelist/
PS C:\Users\disha\OneDrive\Desktop\PROJECT\jobfinder\server> node serve
r.js
```

## Key Directories and Files

1. **/controllers**
   - Contains functions to handle requests and responses.
2. **/models**
   - Includes Mongoose schemas and models for MongoDB collections.
3. **/routes**
   - Defines the API endpoints and links them to controller functions.
4. **/middlewares**
   - Custom middleware functions for request processing.
5. **/config**
   - Configuration files for database connections, environment variables, etc.

**API Endpoints**
A summary of the main API endpoints and their purposes:

**User Authentication**

- **POST /api/user/register** - Registers a new user.
- **POST /api/user/login** - Authenticates a user and returns a token.

**User Management**

- **GET /api/user/-** Retrieves user information by ID.
- **PUT /api/user/**- Updates user information by ID.

**Workout Plans**

- **GET /api/workoutplans** - Retrieves all workout plans.
- **POST /api/workoutplans** - Creates a new workout plan.

**Equipment**

- **GET /api/equipment** - Retrieves all equipment.
- **POST /api/equipment** - Adds new equipment.

**Monthly Plans**

- **GET /api/monthlyplans** - Retrieves all monthly plans.
- **POST /api/monthlyplans** - Creates a new monthly plan.

**Integration with Frontend**
The backend communicates with the frontend via RESTful APIs. Key points of integration include:

- **User Authentication:** Tokens are passed between frontend and backend to handle authentication.
- **Data Fetching:** Frontend components make API calls to fetch necessary data for display and interaction.

**Error Handling and Validation**
Describe the error handling strategy and validation mechanisms:

- **Error Handling:** Centralized error handling using middleware.
- **Validation:** Input validation using libraries like Joi or express-validator.

**Security Considerations**
Outline the security measures implemented:

- **Authentication:** Secure token-based authentication.
- **Data Encryption:** Encrypt sensitive data at rest and in transit.