

## TCH – PIS Code Document

### Code Directory Structure

#### TCH – PIS /

```
|— .env
|— .gitignore
|— Dockerfile
|— Jenkinsfile
|— package.json
|— package-lock.json
|— sonar-project.properties
|— shared/
|   |— db.js
|   |— counterModel.js
|— services/
|   |— auth-service/
|   |   |— src/
|   |   |   |— controllers/
|   |   |   |   |— authController.js
|   |   |   |   |— models/
|   |   |   |   |   |— User.js
|   |   |   |   |— routes/
|   |   |   |   |   |— authRoutes.js
|   |   |   |   |— server.js
|   |   |— package.json
|   |   |— test/
|   |   |   |— authControllerUnit.test.js
|   |— patient-registration/
|   |   |— src/
|   |   |   |— controllers/
|   |   |   |   |— patientController.js
|   |   |   |   |— models/
|   |   |   |   |   |— Patient.js
|   |   |   |   |— routes/
|   |   |   |   |   |— patientRoutes.js
|   |   |   |   |— server.js
|   |   |— test/
|   |   |   |— patient-registration.test.js
|   |— patient-treatment/
|   |   |— src/
|   |   |   |— controllers/
|   |   |   |   |— treatmentController.js
|   |   |   |   |— models/
|   |   |   |   |   |— Treatment.js
|   |   |   |   |— routes/
|   |   |   |   |   |— treatmentRoutes.js
|   |   |   |   |— server.js
|   |   |— test/
|   |   |   |— patienttreatmentUnit.test.js
```

## CODE Files

---

### 1. Core Configuration

#### a. TCH – PIS – Main / .env –

```
MONGO_URI=mongodb+srv://fmc4000:PcMk4CYVjrNQ9LW8@patient-info-  
system.qj5ku.mongodb.net/patientdb?retryWrites=true&w=majority  
JWT_SECRET=e5b8a6d8f92c  
#PORT=3000
```

---

#### b. TCH – PIS – Main / .gitignore –

```
node_modules/
```

---

#### c. TCH – PIS – Main / Dockerfile –

```
# Use Node.js LTS as the base image  
FROM node:18-alpine  
  
# Set the working directory inside the container  
WORKDIR /app  
  
# Copy package.json and package-lock.json to install dependencies  
COPY package.json package-lock.json ./  
  
# Install dependencies for the monorepo  
RUN npm install  
  
# Copy the rest of the application files  
COPY . .  
  
# Expose ports used by each service (adjust if necessary)  
EXPOSE 3000 3001 3002  
  
# Command to run all services concurrently  
CMD ["npm", "run", "start-all"]
```

d. TCH – PIS – Main / Jenkinsfile

```
pipeline {
  agent any

  stages {
    stage('Launch Docker Container') {
      steps {
        script {
          // Run the container in detached mode with port mappings
          sh '''
            docker run -d --name tch-pis-container \
            -p 3000:3000 -p 3001:3001 -p 3002:3002 \
            tch-pis-image:1.0 tail -f /dev/null
          '''
        }
      }
    }

    stage('UNIT_TEST') {
      steps {
        script {
          // Clone the repo inside the running container and install dependencies
          sh '''
            docker exec tch-pis-container git clone https://github.com/faisalmc/TCH-PIS.git /app/TCH-PIS
            docker exec tch-pis-container sh -c "cd /app/TCH-PIS && npm install"
            docker exec tch-pis-container sh -c "cd /app/TCH-PIS && npm run test"
          '''
        }
      }
    }

    stage('Static Code Analysis') {
      steps {
        script {
          // 'SonarQube' is the identifier for the SonarQube server configured in Jenkins global settings
          withSonarQubeEnv('SonarQube') {
            sh 'sonar-scanner'
          }
        }
      }
    }

    stage('BUILD') {
      when {
        expression {
          // Run BUILD stage only if UNIT_TEST is successful
          currentBuild.result == null || currentBuild.result == 'SUCCESS'
        }
      }
      steps {
        script {
          // Run start-all command inside the container
          sh '''
            docker exec tch-pis-container sh -c "cd /app/TCH-PIS && npm run start-all &"
          '''
        }
      }
    }
  }

  post {
    always {
      script {
        // Cleanup: Stop and remove the container after the pipeline
        sh '''
          docker stop tch-pis-container
          docker rm tch-pis-container
        '''
      }
    }
  }
}
```

e. TCH – PIS – Main / package.json

```
{
  "name": "patient-information-system",
  "version": "1.0.0",
  "private": true,
  "workspaces": [
    "services/*"
  ],
  "scripts": {
    "start": "npm run start --workspace=auth-service",
    "sonar": "sonar-scanner",
    "start-all": "concurrently \"node services/auth-service/src/server.js\" \"node services/patient-registration/src/server.js\" \"node services/patient-treatment/src/server.js\"",
    "test": "cd services/auth-service && npm test && cd ../patient-registration && npm test && cd ../patient-treatment && npm test"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "chai": "^5.2.0",
    "mocha": "^11.1.0",
    "nodemon": "^3.1.9",
    "sinon": "^19.0.2"
  },
  "description": "Monorepo for Patient Information System",
  "dependencies": {
    "concurrently": "^9.1.2",
    "express": "^4.21.2"
  }
}
```

---

f. TCH – PIS – Main / package-lock.json

```
{
  "name": "patient-information-system",
  "version": "1.0.0",
  "lockfileVersion": 3,
  "requires": true,
  "packages": {
    "": {
      "name": "patient-information-system",
      "version": "1.0.0",
      "license": "ISC",
      "workspaces": [
        "services/*"
      ],
      "dependencies": {
        "express": "^4.21.2"
      },
      "devDependencies": {
        "chai": "^5.2.0",
        "mocha": "^11.1.0",
        "nodemon": "^3.1.9",
        "sinon": "^19.0.2"
      }
    },
    "node_modules/@isaacs/cliui": {
      "version": "8.0.2",
      "resolved": "https://registry.npmjs.org/@isaacs/cliui/-/cliui-8.0.2.tgz",
      "integrity": "sha512-O8jcjabXaleOG9DQ0+ARXWZBTfnP4WNAqzuiJK7ll44AmxGKv/J2M4TPjxjY3znBCfvBXFzucm1twdyFybFqEA==",
      "dev": true,
      "license": "ISC",
      "dependencies": {
        "string-width": "^5.1.2",
        "string-width-cjs": "npm:string-width@^4.2.0",
        "strip-ansi": "^7.0.1",
        "strip-ansi-cjs": "npm:strip-ansi@^6.0.1",
        "wrap-ansi": "^8.1.0",
        "wrap-ansi-cjs": "npm:wrap-ansi@^7.0.0"
      },
      "engines": {
        "node": ">=12"
      }
    },
    "node_modules/@isaacs/cliui/node_modules/ansi-regex": {
      "version": "6.1.0",
      "resolved": "https://registry.npmjs.org/ansi-regex/-/ansi-regex-6.1.0.tgz",
      "integrity": "sha512-7YXfoYUhyRw/+pnb0XdbOJMIa1VfJj+9oOHd0B5vkGjL9fEI4BnI5/20yvkWjWq55yRumP09FpB9X8605g==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=12"
      }
    },
    "node_modules/@isaacs/cliui/node_modules/ansi-styles": {
      "version": "6.2.1",
      "resolved": "https://registry.npmjs.org/ansi-styles/-/ansi-styles-6.2.1.tgz",
      "integrity": "sha512-9806EayG9Hj5b36I5sTdk19LxzREuRSG6Z9Rr68z4O7Lxf462Va5YqulMlX8dfUwWu6B883wZ0rrR8l8Zg==",
      "dev": true,
      "license": "MIT",
      "engines": {
        "node": ">=12"
      }
    },
    "node_modules/@isaacs/cliui/node_modules/emoji-regex": {
      "version": "9.2.2",
      "resolved": "https://registry.npmjs.org/emoji-regex/-/emoji-regex-9.2.2.tgz",
      "integrity": "sha512-l5nT4Dd2LmB47hd/jvvc63zOH38qyZkLqS03++870LxSjU7ZK+h8p2IhWzFShPt8H9Kz9Tq2j5I+V2XUAA==",
      "dev": true,
      "license": "MIT"
    },
    "node_modules/@isaacs/cliui/node_modules/string-width": {
      "version": "5.1.2",

```

```

"resolved": "https://registry.npmjs.org/string-width/-/string-width-5.1.2.tgz",
"integrity": "sha512-",
HnLOCR3vjY8beoNLtcjZ5/nxn2afmME6lhrDrebokqMap+XbeW8n9TXpPDOqdGK5qcl3oT0GKtW6wC7EMIVqA==",
"dev": true,
"license": "MIT",
"dependencies": {
"eastasianwidth": "^0.2.0",
"emoji-regex": "^9.2.2",
"strip-ansi": "^7.0.1"
},
"engines": {
"node": ">=12"
},
"funding": {
"url": "https://github.com/sponsors/sindresorhus"
}
},
"node_modules/@isaacs/cliui/node_modules/strip-ansi": {
"version": "7.1.0",
"resolved": "https://registry.npmjs.org/strip-ansi/-/strip-ansi-7.1.0.tgz",
"integrity": "sha512-",
iq6eVVI64nQQTRYq2KtEg2d2uU7LElhTJwsH4YzIHZshxlgZms/wlc4VoDQTLG/lvVlrBKG06CrZnp0qv7hkcQ==",
"dev": true,
"license": "MIT",
"dependencies": {
"ansi-regex": "^6.0.1"
},
"engines": {
"node": ">=12"
},
"funding": {
"url": "https://github.com/chalk/strip-ansi?sponsor=1"
}
},
"node_modules/@isaacs/cliui/node_modules/wrap-ansi": {
"version": "8.1.0",
"resolved": "https://registry.npmjs.org/wrap-ansi/-/wrap-ansi-8.1.0.tgz",
"integrity": "sha512-",
si7QWI6zUMq56bESFvagtzmzMdGotoxfR+Sez11Mobfc7tm+VkJckk9bW2UeffTGVUbOksxmSw0AA2gs8g71NCQ==",
"dev": true,
"license": "MIT",
"dependencies": {
"ansi-styles": "^6.1.0",
"string-width": "^5.0.1",
"strip-ansi": "^7.0.1"
},
"engines": {
"node": ">=12"
},
"funding": {
"url": "https://github.com/chalk/wrap-ansi?sponsor=1"
}
},
"node_modules/@mapbox/node-pre-gyp": {
"version": "1.0.11",
"resolved": "https://registry.npmjs.org/@mapbox/node-pre-gyp/-/node-pre-gyp-1.0.11.tgz",
"integrity": "sha512-",
Yhlar6v9WQgUp/He7BdgzOz8lqMQ8sU+jkCq7Wx8Myc5YFJLbEe7lgui/V7G1qB1DJykHSGwreceSaD60Y0PUQ==",
"license": "BSD-3-Clause",
"dependencies": {
"detect-libc": "^2.0.0",
"https-proxy-agent": "^5.0.0",
"make-dir": "^3.1.0",
"node-fetch": "^2.6.7",
"nopt": "^5.0.0",
"npmlog": "^5.0.1",
"rimraf": "^3.0.2",
"semver": "^7.3.5",
"tar": "^6.1.11"
},
"bin": {
"node-pre-gyp": "bin/node-pre-gyp"
}
},
"node_modules/@mongodb-js/saslprep": {
"version": "1.1.9",
"resolved": "https://registry.npmjs.org/@mongodb-js/saslprep/-/saslprep-1.1.9.tgz",
"integrity": "sha512-",
tvKljjeEaAhCqTzajSdgbQ6gE6f3oneVwa3iXR6csiEwXXOFsCi6Uh9iAjAhXPtqa/XMDHWjjeNH/77m/Yq2dw==",
"license": "MIT",
"dependencies": {

```

```
"sparse-bitfield": "^3.0.3"
}
},
"node_modules/@pkgjs/parseargs": {
  "version": "0.11.0",
  "resolved": "https://registry.npmjs.org/@pkgjs/parseargs/-/parseargs-0.11.0.tgz",
  "integrity": "sha512-1VkdD0QBLPodGrJUEqarH8VAlvQODlbwh9XpP5Syisf7YoQgsJKPNFoqqLQlu+VQ/tVSshMR6loPMn8U+dPg==",
  "dev": true,
  "license": "MIT",
  "optional": true,
  "engines": {
    "node": ">=14"
  }
},
"node_modules/@sinonjs/commons": {
  "version": "3.0.1",
  "resolved": "https://registry.npmjs.org/@sinonjs/commons/-/commons-3.0.1.tgz",
  "integrity": "sha512-K3mCHKQ9sVh8o1C9cxkwxoOmXoAMlDxC1mYyHrjqOWEcBjYr76t96zL2zlj5dUGZ3HSw240X1qgH3MjfyJWpQ==",
  "dev": true,
  "license": "BSD-3-Clause",
  "dependencies": {
    "type-detect": "4.0.8"
  }
},
"node_modules/@sinonjs/fake-timers": {
  "version": "13.0.5",
  "resolved": "https://registry.npmjs.org/@sinonjs/fake-timers/-/fake-timers-13.0.5.tgz",
  "integrity": "sha512-36/hTbH2uaWuGVERyC6da9YwGWnzUZXuPro/F2fsdOsLnCojz/iSH8MxUt/FD2S5XBSVPhmArFUXcpCQ2Hkiw==",
  "dev": true,
  "license": "BSD-3-Clause",
  "dependencies": {
    "@sinonjs/commons": "^3.0.1"
  }
},
"node_modules/@sinonjs/samsam": {
  "version": "8.0.2",
  "resolved": "https://registry.npmjs.org/@sinonjs/samsam/-/samsam-8.0.2.tgz",
  "integrity": "sha512-v46t/fwnhejRSFTGqbnp9u+LQ9xJDse10gNnPgAcxgdoCDMXj/G2asWAC/8Qs+BAZDicX+MNZouXT1A7c83kVw==",
  "dev": true,
  "license": "BSD-3-Clause",
  "dependencies": {
    "@sinonjs/commons": "^3.0.1",
    "lodash.get": "^4.4.2",
    "type-detect": "^4.1.0"
  }
},
"node_modules/@sinonjs/samsam/node_modules/type-detect": {
  "version": "4.1.0",
  "resolved": "https://registry.npmjs.org/type-detect/-/type-detect-4.1.0.tgz",
  "integrity": "sha512-59Y4gvQ6M1LXJfG6YnjWcg3Rqd2611WY2YU08GgK9F7uqHdzsTiO521L3O1J2cu0PLtLzU8iKzV48Yw==",
  "dev": true,
  "license": "MIT",
  "engines": {
    "node": ">=4"
  }
},
"node_modules/@sinonjs/text-encoding": {
  "version": "0.7.3",
  "resolved": "https://registry.npmjs.org/@sinonjs/text-encoding/-/text-encoding-0.7.3.tgz",
  "integrity": "sha512-4s7e1tSdJ30TbL5TzBz1Aa8Tzjw5d980zKwD05geozKZuWcA4PgncdV+E2bWqMAiuF7z/z7J2Z0z/yw==",
  "dev": true,
  "license": "(Unlicense OR Apache-2.0)"
},
"node_modules/@types/webidl-conversions": {
  "version": "7.0.3",
  "resolved": "https://registry.npmjs.org/@types/webidl-conversions/-/webidl-conversions-7.0.3.tgz",
  "integrity": "sha512-4s7e1tSdJ30TbL5TzBz1Aa8Tzjw5d980zKwD05geozKZuWcA4PgncdV+E2bWqMAiuF7z/z7J2Z0z/yw==",
  "license": "MIT"
},
"node_modules/@types/whatwg-url": {
  "version": "11.0.5",
  "resolved": "https://registry.npmjs.org/@types/whatwg-url/-/whatwg-url-11.0.5.tgz",
  "integrity": "sha512-coYR071JRaHa+xEvvYqvnIHAvQaYrLPbsufM9BF63HkwI5Lgmy2QR8Q5K/IYDY05AK82wOvSOS0UsLTpTG7uQ==",
```

```
"license": "MIT",
"dependencies": {
  "@types/webidl-conversions": "*"
}
},
"node_modules/abbrev": {
  "version": "1.1.1",
  "resolved": "https://registry.npmjs.org/abbrev/-/abbrev-1.1.1.tgz",
  "integrity": "sha512-nne9/hzlhy6pdDnbBtz7DjPTKrY00P/zvPSm5pOFkl6xuGrGnXn/VtTNNfNtAfZ9/1RtehkszU9qcTii0Q==",
  "license": "ISC"
},
"node_modules/accepts": {
  "version": "1.3.8",
  "resolved": "https://registry.npmjs.org/accepts/-/accepts-1.3.8.tgz",
  "integrity": "sha512-DN0MX3vCBM8F11V8h6cUfXJW8jBhfREYU9fL6KLt7G8rIrdG9H0YM7ZQHEDvYxkSmQ9tVxeAtpnB0UQ==",
  "license": "MIT",
  "dependencies": {
    "mime-types": "~2.1.34",
    "negotiator": "0.6.3"
  },
  "engines": {
    "node": ">= 0.6"
  }
},
"node_modules/agent-base": {
  "version": "6.0.2",
  "resolved": "https://registry.npmjs.org/agent-base/-/agent-base-6.0.2.tgz",
  "integrity": "sha512-RZNwNclIF7+MS/8bDg70amg32dyeZGXiDuQmZxKLALQjr3jGyLx+4Kkk58UO7D2QdgFIQCovuSuZESne6RG6XQ==",
  "license": "MIT",
  "dependencies": {
    "debug": "4"
  },
  "engines": {
    "node": ">= 6.0.0"
  }
},
"node_modules/agent-base/node_modules/debug": {
  "version": "4.4.0",
  "resolved": "https://registry.npmjs.org/debug/-/debug-4.4.0.tgz",
  "integrity": "sha512-626WE5oiaYVIsS15NvBg7+i0e1g1q4G4j4w4YaXqYkFq80z6V4Cn7o9N1fYpSO1VqL6L4+YU8oQ==",
  "license": "MIT",
  "dependencies": {
    "ms": "^2.1.3"
  },
  "engines": {
    "node": ">=6.0"
  },
  "peerDependenciesMeta": {
    "supports-color": {
      "optional": true
    }
  }
},
"node_modules/agent-base/node_modules/ms": {
  "version": "2.1.3",
  "resolved": "https://registry.npmjs.org/ms/-/ms-2.1.3.tgz",
  "integrity": "sha512-q8j58U0Qv7X16bS9eNiqbU94uX00Y8VW8Ez8j5dT9ZLW6Lz6YKLQEN/wZ8v6Na2sI2ZuI1d88oQ==",
  "license": "MIT"
},
"node_modules/ansi-colors": {
  "version": "4.1.3",
  "resolved": "https://registry.npmjs.org/ansi-colors/-/ansi-colors-4.1.3.tgz",
  "integrity": "sha512-P7UQtdV69Yq8v5e4M4UaogeuGK19CvLfV243ZFq80NteIEYtW5H1eQad4PQwy8StZosxbCZ8FroQ==",
  "dev": true,
  "license": "MIT",
  "engines": {
    "node": ">=6"
  }
},
"node_modules/ansi-regex": {
  "version": "5.0.1",
  "resolved": "https://registry.npmjs.org/ansi-regex/-/ansi-regex-5.0.1.tgz",
  "integrity": "sha512-UJw09AJEFMyjS6kxVVAy70R9tG69P97dBZ8q9s7qSDA0taupH0kXwTw9JtL1pMI5fMD08IAH836oQ==",
  "license": "MIT",
  "engines": {
    "node": ">=8"
  }
}
```

```
}
},
"node_modules/ansi-styles": {
  "version": "4.3.0",
  "resolved": "https://registry.npmjs.org/ansi-styles/-/ansi-styles-4.3.0.tgz",
  "integrity": "sha512-7Bz2329beB33e330dC7Cai3eZEF7758hhYb1eZa894b4P2e2S9l86C7G476C63935638B78C866731",
  "dev": true,
  "license": "MIT",
  "dependencies": {
    "color-convert": "^2.0.1"
  },
  "engines": {
    "node": ">=8"
  },
  "funding": {
    "url": "https://github.com/chalk/ansi-styles?sponsor=1"
  },
  "node_modules/anymatch": {
    "version": "3.1.3",
    "resolved": "https://registry.npmjs.org/anymatch/-/anymatch-3.1.3.tgz",
    "integrity": "sha512-KMReFUr0B4t+D+OBkjr3KYqvocp2XaSzO55UcB6mgQMd3KbcE+mWTyvVV7D/zsdEbNnV6acZUutkiHQXvTr1Rw==",
    "dev": true,
    "license": "ISC",
    "dependencies": {
      "normalize-path": "^3.0.0",
      "picomatch": "^2.0.4"
    },
    "engines": {
      "node": ">= 8"
    }
  },
  "node_modules/aproba": {
    "version": "2.0.0",
    "resolved": "https://registry.npmjs.org/aproba/-/aproba-2.0.0.tgz",
    "integrity": "sha512-lh4EjI64a5JdAuYt2a0wZGKw6qAr1F17V6O2U5SsUxqYvFV4Sdh5yc2bk2P0yN3F0z4qF0a59a5uZuzrQ==",
    "license": "ISC"
  },
  "node_modules/are-we-there-yet": {
    "version": "2.0.0",
    "resolved": "https://registry.npmjs.org/are-we-there-yet/-/are-we-there-yet-2.0.0.tgz",
    "integrity": "sha512-2uohvW30xJ1Y8Dxz4Ywq+Bz9xAYEg1YJ18I0dFb8S0i6W0pZT8T0F5X+pEGzjkOH31Fks0O1SY1TJ8qA==",
    "deprecated": "This package is no longer supported.",
    "license": "ISC",
    "dependencies": {
      "delegates": "^1.0.0",
      "readable-stream": "^3.6.0"
    },
    "engines": {
      "node": ">=10"
    }
  },
  "node_modules/argparse": {
    "version": "2.0.1",
    "resolved": "https://registry.npmjs.org/argparse/-/argparse-2.0.1.tgz",
    "integrity": "sha512-8+9WqebbFzpX9OR+Wa6O29aslogeRMzcGtAlNdpMHHyAg10f05aSFVBbcEqGf/PXw1EjAZ+q2/bEBg3DvurK3Q==",
    "dev": true,
    "license": "Python-2.0"
  },
  "node_modules/array-flatten": {
    "version": "1.1.1",
    "resolved": "https://registry.npmjs.org/array-flatten/-/array-flatten-1.1.1.tgz",
    "integrity": "sha512-3NN8m8V5GfU1jKnR8Z19vJ8cFl7rJ1cPE5C1gWd8hag61YRjOvF2Ihw2YH8B28ln1QYd/hF6nPUZqR4L6w==",
    "license": "MIT"
  },
  "node_modules/assertion-error": {
    "version": "2.0.1",
    "resolved": "https://registry.npmjs.org/assertion-error/-/assertion-error-2.0.1.tgz",
    "integrity": "sha512-IjybGOEA8xJ1AiF/o6WAPbVZ/HkEiNtZlWlUcK7N9Fb/jksHqzPQQmAW5U488qZgixh0QmEgZaXufD8R6wg==",
    "dev": true,
    "license": "MIT",
    "engines": {
      "node": ">=12"
    }
  },
}
```

```
"node_modules/auth-service": {
  "resolved": "services/auth-service",
  "link": true
},
"node_modules/balanced-match": {
  "version": "1.0.2",
  "resolved": "https://registry.npmjs.org/balanced-match/-/balanced-match-1.0.2.tgz",
  "integrity": "sha512-3oSeU00TMV67hN1AmbXsK4yaqU77jiHlbbRDZOpH0KW9+CeX4bRAaX0Anxt0tx2MrpRpWwQaPwllIIEJhYU5Pw==",
  "license": "MIT"
},
"node_modules/bcrypt": {
  "version": "5.1.1",
  "resolved": "https://registry.npmjs.org/bcrypt/-/bcrypt-5.1.1.tgz",
  "integrity": "sha512-HbSx4wB7ZEbrVMwxhYxOAsQ8kIXGA6QpbbXfn6E24G4LmMfB3HErYYyNJyS/TpbyZSTBDEzYgR5XwFFsgF6aQ==",
  "hasInstallScript": true,
  "license": "MIT",
  "dependencies": {
    "@mapbox/node-pre-gyp": "^1.0.11",
    "node-addon-api": "^5.0.0"
  },
  "engines": {
    "node": ">= 10.0.0"
  }
},
"node_modules/bcryptjs": {
  "version": "2.4.3",
  "resolved": "https://registry.npmjs.org/bcryptjs/-/bcryptjs-2.4.3.tgz",
  "integrity": "sha512-VHy/X9Vt7f3BbPJEi8BdVFMBYHi+jNXrYkW3huaybV/kQ0KJg0Y6PkEMbn+zeT+i+SiKZ/HMqJGIIt4LZDqNQ==",
  "license": "MIT"
},
"node_modules/binary-extensions": {
  "version": "2.3.0",
  "resolved": "https://registry.npmjs.org/binary-extensions/-/binary-extensions-2.3.0.tgz",
  "integrity": "sha512-JoLw8Jph8qf8kLJW2e8Tq38YUeB9D71w0qRQxgNT1vwwtYz2a9e2g6J7PmWXlVlH4kU8UzHfBMfw/7ecA==",
  "dev": true,
  "license": "MIT",
  "engines": {
    "node": ">=8"
  },
  "funding": {
    "url": "https://github.com/sponsors/sindresorhus"
  }
},
"node_modules/body-parser": {
  "version": "1.20.3",
  "resolved": "https://registry.npmjs.org/body-parser/-/body-parser-1.20.3.tgz",
  "integrity": "sha512-Wx01x3bMkVw0DWuAPQmXS27WQjP3mvOy0Lq1N78E5v5UDE/oSzgb0ICBY9L9S6ZnvmY61X489OyUa1k08Dg==",
  "license": "MIT",
  "dependencies": {
    "bytes": "3.1.2",
    "content-type": "~1.0.5",
    "debug": "2.6.9",
    "depd": "2.0.0",
    "destroy": "1.2.0",
    "http-errors": "2.0.0",
    "iconv-lite": "0.4.24",
    "on-finished": "2.4.1",
    "qs": "6.13.0",
    "raw-body": "2.5.2",
    "type-is": "~1.6.18",
    "unpipe": "1.0.0"
  },
  "engines": {
    "node": ">= 0.8",
    "npm": "1.2.8000 || >= 1.4.16"
  }
},
"node_modules/brace-expansion": {
  "version": "1.1.11",
  "resolved": "https://registry.npmjs.org/brace-expansion/-/brace-expansion-1.1.11.tgz",
  "integrity": "sha512-iCuPHDFgrHX7H2vEI/5xpz07zSHB00TugqhmYtVmMO6518mCuRMoOYFldEBl0g187ufozdaHgWKcYFb61qGiA==",
  "license": "MIT",
  "dependencies": {
    "balanced-match": "^1.0.0",
    "concat-map": "0.0.1"
  }
}
```

```

    }
  },
  "node_modules/braces": {
    "version": "3.0.3",
    "resolved": "https://registry.npmjs.org/braces/-/braces-3.0.3.tgz",
    "integrity": "sha512-yQbXgO/OSZVD2IsiLLro+7Hf6Q18EJrKSEsdoMzKePKXct3gvD8oLcOQdlzGupr5Fj+EDe8gO/lxc1BzfMpxvA==",
    "dev": true,
    "license": "MIT",
    "dependencies": {
      "fill-range": "^7.1.1"
    }
  },
  "engines": {
    "node": ">=8"
  }
},
"node_modules/browser-stdout": {
  "version": "1.3.1",
  "resolved": "https://registry.npmjs.org/browser-stdout/-/browser-stdout-1.3.1.tgz",
  "integrity": "sha512-qhAVI1+Av2X7qelOfAlYwXONood6XIZE/fXaBSmW/T5SzLAmCgzi+eiWE7fUvbHaeNBQH13UftjpXxsfLkMpgw==",
  "dev": true,
  "license": "ISC"
},
"node_modules/bson": {
  "version": "6.10.1",
  "resolved": "https://registry.npmjs.org/bson/-/bson-6.10.1.tgz",
  "integrity": "sha512-P92xmHDQJSPKPLHqFxefqMxASNq/aWJMEZugpCjf+AF/pgcUpMMQCg7t7+ewko0/u8AapvF3luf/FoehddEK+sA==",
  "license": "Apache-2.0",
  "engines": {
    "node": ">=16.20.1"
  }
},
"node_modules/buffer-equal-constant-time": {
  "version": "1.0.1",
  "resolved": "https://registry.npmjs.org/buffer-equal-constant-time/-/buffer-equal-constant-time-1.0.1.tgz",
  "integrity": "sha512-zRpUIdDwd/xk6ADqPMATG8vc9VPkck7T07Olx0gnjmJAnHnTVXNQG3vfvWNuiZlkwu9KrKdA1iJKfsfTVxE6NA==",
  "license": "BSD-3-Clause"
},
"node_modules/bytes": {
  "version": "3.1.2",
  "resolved": "https://registry.npmjs.org/bytes/-/bytes-3.1.2.tgz",
  "integrity": "sha512-/Nf7TyzTx6S3yRJObOAV7956r8cr2+Oj8AC5dt8wSP3BQAoeX58NoHyCU8P8zGkNXStjTSi6fzO6F0pBdcYbEg==",
  "license": "MIT",
  "engines": {
    "node": ">= 0.8"
  }
},
"node_modules/call-bind-apply-helpers": {
  "version": "1.0.1",
  "resolved": "https://registry.npmjs.org/call-bind-apply-helpers/-/call-bind-apply-helpers-1.0.1.tgz",
  "integrity": "sha512-BhYE+WDaywFg2TBWYNXAE+8B1ATnThNBqXHP5nQu0jWJdVvY2hvkpyB3qOmtmDePiS5/BDQ8wASEWGMWRG148g==",
  "license": "MIT",
  "dependencies": {
    "es-errors": "^1.3.0",
    "function-bind": "^1.1.2"
  },
  "engines": {
    "node": ">= 0.4"
  }
},
"node_modules/call-bound": {
  "version": "1.0.3",
  "resolved": "https://registry.npmjs.org/call-bound/-/call-bound-1.0.3.tgz",
  "integrity": "sha512-YTd+6wGlNPxSuri7Y6X8tY2dmm12UMH66RpKMhiX6rsk5wXXnYgbUcOt8kiS31/AjfoTOvCsE+w8nZQLQnzHA==",
  "license": "MIT",
  "dependencies": {
    "call-bind-apply-helpers": "^1.0.1",
    "get-intrinsic": "^1.2.6"
  },
  "engines": {
    "node": ">= 0.4"
  },
  "funding": {
    "url": "https://github.com/sponsors/ljharb"
  }
}

```

```

},
"node_modules/camelcase": {
  "version": "6.3.0",
  "resolved": "https://registry.npmjs.org/camelcase/-/camelcase-6.3.0.tgz",
  "integrity": "sha512-Gmy6FhYICy7uOEIzUSbxo2UCDH8owEk996gkbrpsgGtrJLM3J7jGxL9lc7Qwwj4ivOE5AWZWRMecDdF7hqGjFA==",
  "dev": true,
  "license": "MIT",
  "engines": {
    "node": ">=10"
  }
},
"node_modules/chai": {
  "version": "5.2.0",
  "resolved": "https://registry.npmjs.org/chai/-/chai-5.2.0.tgz",
  "integrity": "sha512-mCuXncKKk5iCLhfhwTc0izo0gtEmpz5CtG2y8GiOINBLMVS6v8TMRc5TaLWKS6692m9+dVVfzgeVxR5UxWHTYw==",
  "dev": true,
  "license": "MIT",
  "dependencies": {
    "assertion-error": "^2.0.1",
    "check-error": "^2.1.1",
    "deep-eql": "^5.0.1",
    "loupe": "^3.1.0",
    "pathval": "^2.0.0"
  },
  "engines": {
    "node": ">=12"
  }
},
"node_modules/chalk": {
  "version": "4.1.2",
  "resolved": "https://registry.npmjs.org/chalk/-/chalk-4.1.2.tgz",
  "integrity": "sha512-oKnbnFyRlXpUuez8iBMmyEa4nbj4lOQyuhc/wy9kY7/WVPcwIO9VA668Pu8RkO7+0G76SLROeyw9CpQ061i4mA==",
  "dev": true,
  "license": "MIT",
  "dependencies": {
    "ansi-styles": "^4.1.0",
    "supports-color": "^7.1.0"
  },
  "engines": {
    "node": ">=10"
  }
},
"node_modules/chalk/node_modules/has-flag": {
  "version": "4.0.0",
  "resolved": "https://registry.npmjs.org/has-flag/-/has-flag-4.0.0.tgz",
  "integrity": "sha512-EykJt/Q1KjTWctppgIAgfSO0tKVuZUjhgMr17kqTumMl6Afv3EISleU7qZUzoXDFTAHTDC4NOoG/ZxU3EvlMPQ==",
  "dev": true,
  "license": "MIT",
  "engines": {
    "node": ">=8"
  }
},
"node_modules/chalk/node_modules/supports-color": {
  "version": "7.2.0",
  "resolved": "https://registry.npmjs.org/supports-color/-/supports-color-7.2.0.tgz",
  "integrity": "sha512-1o9239881t104Q1k80qLlx3ooc6Y4W2/jkZykY19j4oQx900p1+b0p1G31535u30Ss9B/7h1l9+/50+q==",
  "dev": true,
  "license": "MIT",
  "dependencies": {
    "has-flag": "^4.0.0"
  },
  "engines": {
    "node": ">=8"
  }
},
"node_modules/check-error": {
  "version": "2.1.1",
  "resolved": "https://registry.npmjs.org/check-error/-/check-error-2.1.1.tgz",
  "integrity": "sha512-333v1133G0424Z000dx98O8K8u31pX+0B7GxeI5GRG2e48iPqkDutFlwI+MxSf8bAiVozexF3II9UDJg==",
  "dev": true,
  "license": "MIT",
  "engines": {
    "node": ">=8"
  }
}

```

```

    "dev": true,
    "license": "MIT",
    "engines": {
      "node": ">= 16"
    }
  },
  "node_modules/chokidar": {
    "version": "3.6.0",
    "resolved": "https://registry.npmjs.org/chokidar/-/chokidar-3.6.0.tgz",
    "integrity": "sha512-7VT13fmjotKpGipCW9JEQAusEPE+Ei8nl6/g4FBAmIm0GOOLMua9NDDo/DWp0ZAxCr3cPq5ZpBqmPAQgDda2Pw==",
    "dev": true,
    "license": "MIT",
    "dependencies": {
      "anymatch": "~3.1.2",
      "braces": "~3.0.2",
      "glob-parent": "~5.1.2",
      "is-binary-path": "~2.1.0",
      "is-glob": "~4.0.1",
      "normalize-path": "~3.0.0",
      "readdirp": "~3.6.0"
    },
    "engines": {
      "node": ">= 8.10.0"
    },
    "funding": {
      "url": "https://paulmillr.com/funding/"
    },
    "optionalDependencies": {
      "fsevents": "~2.3.2"
    }
  },
  "node_modules/chownr": {
    "version": "2.0.0",
    "resolved": "https://registry.npmjs.org/chownr/-/chownr-2.0.0.tgz",
    "integrity": "sha512-8eA1uVrDd+ue1VUd4RQYdNkLjKz8oFyYp8oHMeOZuS58odnwZTRZl6wPCrVlO8QhQFf8z9j4HagHkmVvA==",
    "license": "ISC",
    "engines": {
      "node": ">=10"
    }
  },
  "node_modules/cliui": {
    "version": "8.0.1",
    "resolved": "https://registry.npmjs.org/cliui/-/cliui-8.0.1.tgz",
    "integrity": "sha512-91207S89817291108927ZC00L9960L7UWUJ0KjX4o7Z8RZd5K88598o6W0b0Vt9Z0e84JEuJ8v9w==",
    "dev": true,
    "license": "ISC",
    "dependencies": {
      "string-width": "^4.2.0",
      "strip-ansi": "^6.0.1",
      "wrap-ansi": "^7.0.0"
    },
    "engines": {
      "node": ">=12"
    }
  },
  "node_modules/color-convert": {
    "version": "2.0.1",
    "resolved": "https://registry.npmjs.org/color-convert/-/color-convert-2.0.1.tgz",
    "integrity": "sha512-RRECPsj7iu/xb5oKYcsFHSppFNnsj/52OVTRKb4zP5onXwVF3zVmmToNcOfGC+CRDpfK/U584fMg38ZHCaElKQ==",
    "dev": true,
    "license": "MIT",
    "dependencies": {
      "color-name": "~1.1.4"
    },
    "engines": {
      "node": ">=7.0.0"
    }
  },
  "node_modules/color-name": {
    "version": "1.1.4",
    "resolved": "https://registry.npmjs.org/color-name/-/color-name-1.1.4.tgz",
    "integrity": "sha512-dXyMd8r5uIdIYwy23ewmIz6OUgdW77jV0L5CgI5GdYqpGgWZfUqzgBsCq2kx5Q44EFV464a3lGAwE7Q==",
    "dev": true,
    "license": "MIT"
  },
  "node_modules/color-support": {

```

```
"version": "1.1.3",
"resolved": "https://registry.npmjs.org/color-support/-/color-support-1.1.3.tgz",
"integrity": "sha512-qIBjKpbMLO/HL68y+lh4q0/O1MZfj2RX6X/KmMa3+gJD3z+Wwl1ZzDHysvqHGS3mP6mznPckpXmw1nl9cJjyRg==",
"license": "ISC",
"bin": {
  "color-support": "bin.js"
}
},
"node_modules/concat-map": {
"version": "0.0.1",
"resolved": "https://registry.npmjs.org/concat-map/-/concat-map-0.0.1.tgz",
"integrity": "sha512-sKa/9XKkhBpPgZ3i90Oqe7l8V4cKlvdqFm38X4Y853/Dy4tmZkH1YoD4iYdIRb7HhEh3Q/5UeBJyYpQMg==",
"license": "MIT"
},
"node_modules/console-control-strings": {
"version": "1.1.0",
"resolved": "https://registry.npmjs.org/console-control-strings/-/console-control-strings-1.1.0.tgz",
"integrity": "sha512-ty/fTekppD2flwRvnZAVdeOiGd1c7YXEixbgJTNzqcXJWKQnj/V1bNEEE6hygpM3WjwHFUVK6HTjWSzV4a8sQ==",
"license": "ISC"
},
"node_modules/content-disposition": {
"version": "0.5.4",
"resolved": "https://registry.npmjs.org/content-disposition/-/content-disposition-0.5.4.tgz",
"integrity": "sha512-FveZTNuGw04cxIAiWbzi6zTAL/lhehaWbTtglwJh4/E95DqMwTmha3KZN1aAWA8cFIhHzMZUvLevkw5Rqk+tSQ==",
"license": "MIT",
"dependencies": {
  "safe-buffer": "5.2.1"
}
},
"engines": {
  "node": ">= 0.6"
}
},
"node_modules/content-type": {
"version": "1.0.5",
"resolved": "https://registry.npmjs.org/content-type/-/content-type-1.0.5.tgz",
"integrity": "sha512-nTjqfCBFeipKdXCv4YDQWCfmcLZKm81ldF0pAopTvyrFGVbcR6P/VAA5G7N+0tTr8QqiU0tFadD6FK4NtJwOA==",
"license": "MIT",
"engines": {
  "node": ">= 0.6"
}
},
"node_modules/cookie": {
"version": "0.7.1",
"resolved": "https://registry.npmjs.org/cookie/-/cookie-0.7.1.tgz",
"integrity": "sha512-+Iw9tXZL9W46YU1j293AT08gW96N2E8Dn9S2N/5O6Y81P4u78p1Wn1lW9Y8L6WVz63R+JWVY565W==",
"license": "MIT",
"engines": {
  "node": ">= 0.6"
}
},
"node_modules/cookie-signature": {
"version": "1.0.6",
"resolved": "https://registry.npmjs.org/cookie-signature/-/cookie-signature-1.0.6.tgz",
"integrity": "sha512-QADzlaHc8icV8I7vbaJXJwod9HWYp8uCqf1xa4OfNu1T7JVxQlRUGOWtHdNDtPiywmFbiS12VjotlXLRKM3orQ==",
"license": "MIT"
},
"node_modules/cross-spawn": {
"version": "7.0.6",
"resolved": "https://registry.npmjs.org/cross-spawn/-/cross-spawn-7.0.6.tgz",
"integrity": "sha512-uVzQ0WfqG7F84J64765uG5D4v8zscD80WW30Ck2u6C5rA2pWtWp6tUv8YSkPldL4BNxWk5HUyW9pA==",
"dev": true,
"license": "MIT",
"dependencies": {
  "path-key": "^3.1.0",
  "shebang-command": "^2.0.0",
  "which": "^2.0.1"
}
},
"engines": {
  "node": ">= 8"
}
},
"node_modules/debug": {
"version": "2.6.9",
```

```
"resolved": "https://registry.npmjs.org/debug/-/debug-2.6.9.tgz",
"integrity": "sha512-bC7ElrdJaJnPbAP+1EotYvqZsb3ecl5wi6Bfi6BJTUcNowp6cvspg0jXznRTKDjm/E7AdgFBVeAPVMNcKGsHMA==",
"license": "MIT",
"dependencies": {
  "ms": "2.0.0"
}
},
"node_modules/decamelize": {
  "version": "4.0.0",
  "resolved": "https://registry.npmjs.org/decamelize/-/decamelize-4.0.0.tgz",
  "integrity": "sha512-9E1PgSik9HelIw2JO94lidnE3eBoQrFJ3w7sFuzSX4DpmZ3v5sZpUiV5Swcf6mQEF+Y0ru8Neo+p+nyh2J+hQ==",
  "dev": true,
  "license": "MIT",
  "engines": {
    "node": ">=10"
  },
  "funding": {
    "url": "https://github.com/sponsors/sindresorhus"
  }
},
"node_modules/deep-eql": {
  "version": "5.0.2",
  "resolved": "https://registry.npmjs.org/deep-eql/-/deep-eql-5.0.2.tgz",
  "integrity": "sha512-h5k5U50IJFpzfL6nO9jaaumfjO/f2NjK/oYB2Djzm4p9L+3T9qWpZqZ2hAbLPuuYq9wrU08WQyBTL5GbPk5Q==",
  "dev": true,
  "license": "MIT",
  "engines": {
    "node": ">=6"
  }
},
"node_modules/delegates": {
  "version": "1.0.0",
  "resolved": "https://registry.npmjs.org/delegates/-/delegates-1.0.0.tgz",
  "integrity": "sha512-bd2L678uiWATM6m5Z1VzNCErI3jiGzt6HGY8OVICs40JQq/HALfbyNJmp0UDakEY4pMMaN0Ly5om/B1VI/+xfQ==",
  "license": "MIT"
},
"node_modules/depd": {
  "version": "2.0.0",
  "resolved": "https://registry.npmjs.org/depd/-/depd-2.0.0.tgz",
  "integrity": "sha512-g7Nf75L5qkLugspcMRfoFvL90RuPyFnsZpjiXq5088F5Q1ukOvLGOfIX1aImUulBjzv87A8uJ3BnZSa4Sv5g==",
  "license": "MIT",
  "engines": {
    "node": ">= 0.8"
  }
},
"node_modules/destroy": {
  "version": "1.2.0",
  "resolved": "https://registry.npmjs.org/destroy/-/destroy-1.2.0.tgz",
  "integrity": "sha512-2sJGTAxIIaR1w4iJSNoN0hnMY7Gpc/n8D4qSCJw8QqFWXf7cuAgnEHxBpweaVcPevC2l3KpjYCx3NypQQgaJg==",
  "license": "MIT",
  "engines": {
    "node": ">= 0.8",
    "npm": "1.2.8000 || >= 1.4.16"
  }
},
"node_modules/detect-libc": {
  "version": "2.0.3",
  "resolved": "https://registry.npmjs.org/detect-libc/-/detect-libc-2.0.3.tgz",
  "integrity": "sha512-bwy0MGW55bG41VqxxypOsdSdGqLwXPl/focwgTYCFMbduiBAXLg9CFzG08sz2aqzknwiX7Hk10bQENjg8ILByw==",
  "license": "Apache-2.0",
  "engines": {
    "node": ">=8"
  }
},
"node_modules/diff": {
  "version": "5.2.0",
  "resolved": "https://registry.npmjs.org/diff/-/diff-5.2.0.tgz",
  "integrity": "sha512-IFJJTajzHnfycioLhJ36u8Q4CqY6sLUGbGE9iK1Uy5XQsD9RqH0KXUejvCHx1yKlUeazb1Gzlc2tWzSAbg==",
  "dev": true,
  "license": "BSD-3-Clause",
  "engines": {
    "node": ">=0.3.1"
  }
}
```

```
},
"node_modules/dotenv": {
  "version": "16.4.7",
  "resolved": "https://registry.npmjs.org/dotenv/-/dotenv-16.4.7.tgz",
  "integrity": "sha512-47qPchRCykZC03FhkYAhrrvwU4xDBFij1QPqaarj6mdM/hgUzFPHcpkHJOn3mJAufFeeAxAzeGsr5X0M4k6fLZQ==",
  "license": "BSD-2-Clause",
  "engines": {
    "node": ">=12"
  },
  "funding": {
    "url": "https://dotenvx.com"
  }
},
"node_modules/dunder-proto": {
  "version": "1.0.1",
  "resolved": "https://registry.npmjs.org/dunder-proto/-/dunder-proto-1.0.1.tgz",
  "integrity": "sha512-KiNnDqBQRCXw0MLVhZE9iQHmG68qAVIBg9CqmUYjmQlhgij9U5MFvrqkUL5FbtyyzZuOeOt0zdeRe4UY7ct+A==",
  "license": "MIT",
  "dependencies": {
    "call-bind-apply-helpers": "^1.0.1",
    "es-errors": "^1.3.0",
    "gopd": "^1.2.0"
  },
  "engines": {
    "node": ">= 0.4"
  }
},
"node_modules/eastasianwidth": {
  "version": "0.2.0",
  "resolved": "https://registry.npmjs.org/eastasianwidth/-/eastasianwidth-0.2.0.tgz",
  "integrity": "sha512-1884v1e3kMeZDkYw+Y4LZdSfDprfcZ814dYUlFVYo3F88vWY3vJYnYSR6cY5U1ieSx8W8lYQYkzpe5g==",
  "dev": true,
  "license": "MIT"
},
"node_modules/ecdsa-sig-formatter": {
  "version": "1.0.11",
  "resolved": "https://registry.npmjs.org/ecdsa-sig-formatter/-/ecdsa-sig-formatter-1.0.11.tgz",
  "integrity": "sha512-nagl3RYrbNv6kQkeJlpt6NJZy8twLB/2vtz6yN9Z4vRKHN4/QZJIEbqohALSGwKdnksuY3k5Addp5lg8sVoVcQ==",
  "license": "Apache-2.0",
  "dependencies": {
    "safe-buffer": "^5.0.1"
  }
},
"node_modules/ee-first": {
  "version": "1.1.1",
  "resolved": "https://registry.npmjs.org/ee-first/-/ee-first-1.1.1.tgz",
  "integrity": "sha512-WMwm9LhRUo+WUaRN+vRuETqG89lgZphVSNkdFgeb6sS/E4OrDIN7t48CAewSHXc6C8lefD8KKfr5vY61brQlow==",
  "license": "MIT"
},
"node_modules/emoji-regex": {
  "version": "8.0.0",
  "resolved": "https://registry.npmjs.org/emoji-regex/-/emoji-regex-8.0.0.tgz",
  "integrity": "sha512-MSjYzcWNOA0ewAHpz0MxpYFvwg6vjy1NG3xteoqz644VCo/RPgnr1/GGt+ic3iJTzQ8Eu3TdM14SawnVUmGE6A==",
  "license": "MIT"
},
"node_modules/encodeurl": {
  "version": "2.0.0",
  "resolved": "https://registry.npmjs.org/encodeurl/-/encodeurl-2.0.0.tgz",
  "integrity": "sha512-Q0n0h+1z2+X/37O109FhQDyQ0dJrR1D51ZdS4jJWpV3N8yHhYD9U//uYB0SjZL2+8frRhtoYD==",
  "license": "MIT",
  "engines": {
    "node": ">= 0.8"
  }
},
"node_modules/es-define-property": {
  "version": "1.0.1",
  "resolved": "https://registry.npmjs.org/es-define-property/-/es-define-property-1.0.1.tgz",
  "integrity": "sha512-0a28o3SDDENNg4t8H9pSjNRHgS/fUOiXo3ZcV7QbQ8hpylHnYHB70rRTBksoUWQWcQ1BtLfzYs9BvQM==",
  "license": "MIT",
  "engines": {
    "node": ">= 0.4"
  }
},

```

```
"node_modules/es-errors": {
  "version": "1.3.0",
  "resolved": "https://registry.npmjs.org/es-errors/-/es-errors-1.3.0.tgz",
  "integrity": "sha512-  
Zf5H2Kxt2xjTvbJvP2ZWLEICxA6j+hAmMzIllypy4xcBg1vKVnx89Wy0GbS+kf5cwCVFFzdCFh2XSCFNULS6csw==",
  "license": "MIT",
  "engines": {
    "node": ">= 0.4"
  }
},
"node_modules/es-object-atoms": {
  "version": "1.1.1",
  "resolved": "https://registry.npmjs.org/es-object-atoms/-/es-object-atoms-1.1.1.tgz",
  "integrity": "sha512-  
FCgH2h8zKNim9lj7dankFPcCIK9Cp5bm+c2gQSYePhpaG5+esrLODihlom+Pe6FGJzWhXQotPv73jTaldXA==",
  "license": "MIT",
  "dependencies": {
    "es-errors": "^1.3.0"
  },
  "engines": {
    "node": ">= 0.4"
  }
},
"node_modules/escalade": {
  "version": "3.2.0",
  "resolved": "https://registry.npmjs.org/escalade/-/escalade-3.2.0.tgz",
  "integrity": "sha512-  
FCgH2h8zKNim9lj7dankFPcCIK9Cp5bm+c2gQSYePhpaG5+esrLODihlom+Pe6FGJzWhXQotPv73jTaldXA==",
  "dev": true,
  "license": "MIT",
  "engines": {
    "node": ">=6"
  }
},
"node_modules/escape-html": {
  "version": "1.0.3",
  "resolved": "https://registry.npmjs.org/escape-html/-/escape-html-1.0.3.tgz",
  "integrity": "sha512-  
TtpcNJ3XAzz3Gq8sWRzJaVajRs0uVxAYakdb1jm2YkPz4G6egUFAYa3n5vtElZefPk5Wa4UXbKuS5fKkJWdgA==",
  "license": "MIT"
},
"node_modules/escape-string-regexp": {
  "version": "4.0.0",
  "resolved": "https://registry.npmjs.org/escape-string-regexp/-/escape-string-regexp-4.0.0.tgz",
  "integrity": "sha512-  
TtpcNJ3XAzz3Gq8sWRzJaVajRs0uVxAYakdb1jm2YkPz4G6egUFAYa3n5vtElZefPk5Wa4UXbKuS5fKkJWdgA==",
  "dev": true,
  "license": "MIT",
  "engines": {
    "node": ">=10"
  },
  "funding": {
    "url": "https://github.com/sponsors/sindresorhus"
  }
},
"node_modules/etag": {
  "version": "1.8.1",
  "resolved": "https://registry.npmjs.org/etag/-/etag-1.8.1.tgz",
  "integrity": "sha512-  
alL5Fx7mawVa300aL2BnEE4iNvo1qETxLrPI/o05L7z6go7fCw1J6EQmbK4FmJ2AS7kgVF/KEZWufBfdCImcPg==",
  "license": "MIT",
  "engines": {
    "node": ">= 0.6"
  }
},
"node_modules/express": {
  "version": "4.21.2",
  "resolved": "https://registry.npmjs.org/express/-/express-4.21.2.tgz",
  "integrity": "sha512-  
28HqgMZAmyh1Czt9ny7qr6ek2qddF4FcIbMzwhCREB0OffH+rXAnuNCwo1/wFvrtbgsQDb4kSbX9de9lFbrXnA==",
  "license": "MIT",
  "dependencies": {
    "accepts": "~1.3.8",
    "array-flatten": "1.1.1",
    "body-parser": "1.20.3",
    "content-disposition": "0.5.4",
    "content-type": "~1.0.4",
    "cookie": "0.7.1",
    "cookie-signature": "1.0.6",
    "debug": "2.6.9",
    "depd": "2.0.0",
    "encodeurl": "~2.0.0",
    "escape-html": "~1.0.3",

```

```

    "etag": "~1.8.1",
    "finalhandler": "1.3.1",
    "fresh": "0.5.2",
    "http-errors": "2.0.0",
    "merge-descriptors": "1.0.3",
    "methods": "~1.1.2",
    "on-finished": "2.4.1",
    "parseurl": "~1.3.3",
    "path-to-regexp": "0.1.12",
    "proxy-addr": "~2.0.7",
    "qs": "6.13.0",
    "range-parser": "~1.2.1",
    "safe-buffer": "5.2.1",
    "send": "0.19.0",
    "serve-static": "1.16.2",
    "setprototypeof": "1.2.0",
    "statuses": "2.0.1",
    "type-is": "~1.6.18",
    "utils-merge": "1.0.1",
    "vary": "~1.1.2"
  },
  "engines": {
    "node": ">= 0.10.0"
  },
  "funding": {
    "type": "opencollective",
    "url": "https://opencollective.com/express"
  }
},
"node_modules/fill-range": {
  "version": "7.1.1",
  "resolved": "https://registry.npmjs.org/fill-range/-/fill-range-7.1.1.tgz",
  "integrity": "sha512-YsGpe3WHLK8Zyi4tWDg2Jy3ebRz2rXowDxnld4bkQB00cc/1Zw9AWnC0i9ztDJitvtQval9KaLyKrc+hBW0yg==",
  "dev": true,
  "license": "MIT",
  "dependencies": {
    "to-regex-range": "^5.0.1"
  },
  "engines": {
    "node": ">=8"
  }
},
"node_modules/finalhandler": {
  "version": "1.3.1",
  "resolved": "https://registry.npmjs.org/finalhandler/-/finalhandler-1.3.1.tgz",
  "integrity": "sha512-6BN9trH7bp3qvnRyZsBz+g3LzXTNZTbVO2EV1CS0WlCdbawYVdYvGfIME/9QP0h0pYlCDBCTjY9nZzMDpyxQ==",
  "license": "MIT",
  "dependencies": {
    "debug": "2.6.9",
    "encodeurl": "~2.0.0",
    "escape-html": "~1.0.3",
    "on-finished": "2.4.1",
    "parseurl": "~1.3.3",
    "statuses": "2.0.1",
    "unpipe": "~1.0.0"
  },
  "engines": {
    "node": ">= 0.8"
  }
},
"node_modules/find-up": {
  "version": "5.0.0",
  "resolved": "https://registry.npmjs.org/find-up/-/find-up-5.0.0.tgz",
  "integrity": "sha512-68Yp5L39iE9NaoiR8x18333B0R6CtYnLr+DvH17YkBg8BpJy7ThfQgA2H1Z0cEjLpBymteci/PkIF0Pd==",
  "dev": true,
  "license": "MIT",
  "dependencies": {
    "locate-path": "^6.0.0",
    "path-exists": "^4.0.0"
  },
  "engines": {
    "node": ">=10"
  },
  "funding": {
    "url": "https://github.com/sponsors/sindresorhus"
  }
},
"node_modules/flat": {

```

```
"version": "5.0.2",
"resolved": "https://registry.npmjs.org/flat/-/flat-5.0.2.tgz",
"integrity": "sha512-b6suED+5/3rTpUBdG1guplI8MPFCAMA0QXwmljLhvCUKcUvdE4gWky9zpuGCcXHOsz4J9wPGNWq6OKpmlzz3hQ==",
"dev": true,
"license": "BSD-3-Clause",
"bin": {
  "flat": "cli.js"
}
},
"node_modules/foreground-child": {
"version": "3.3.1",
"resolved": "https://registry.npmjs.org/foreground-child/-/foreground-child-3.3.1.tgz",
"integrity": "sha512-gIXJkQtFuWegzFRJA9WCqeSJLZDjgJUOMCMzxtvFq/37KojM1BFGufqsCy0r4qSQmYLsZYMeyRqzIWOMup03sw==",
"dev": true,
"license": "ISC",
"dependencies": {
  "cross-spawn": "^7.0.6",
  "signal-exit": "^4.0.1"
}
},
"engines": {
  "node": ">=14"
}
},
"funding": {
  "url": "https://github.com/sponsors/isaacs"
}
},
"node_modules/foreground-child/node_modules/signal-exit": {
"version": "4.1.0",
"resolved": "https://registry.npmjs.org/signal-exit/-/signal-exit-4.1.0.tgz",
"integrity": "sha512-bzyZ1e88w9O1iNjbKnOlvyTrWPDl46O1bG0D3XInv+9tkPrxrN8jUUTiFIDkkmKWgn1M6CfIA13SuGqOa9Korw==",
"dev": true,
"license": "ISC",
"engines": {
  "node": ">=14"
}
},
"funding": {
  "url": "https://github.com/sponsors/isaacs"
}
},
"node_modules/forwarded": {
"version": "0.2.0",
"resolved": "https://registry.npmjs.org/forwarded/-/forwarded-0.2.0.tgz",
"integrity": "sha512-buRG0fpBtrHSTCOASe6hD258tEubFoRLb4ZNA6NxMVHNw2gOcwHo9wyablzMzOA5z9xA9L1KNjk/Nt6MT9aYow==",
"license": "MIT",
"engines": {
  "node": ">= 0.6"
}
},
"node_modules/fresh": {
"version": "0.5.2",
"resolved": "https://registry.npmjs.org/fresh/-/fresh-0.5.2.tgz",
"integrity": "sha512-twkvbTMw88r01M6WVjYwGKv2iFIzWstCzD+BtW5Wu8WwLHD28YwBd7fO24g+8M9TuzBqY4wWv8vGg==",
"license": "MIT",
"engines": {
  "node": ">= 0.6"
}
},
"node_modules/fs-minipass": {
"version": "2.1.0",
"resolved": "https://registry.npmjs.org/fs-minipass/-/fs-minipass-2.1.0.tgz",
"integrity": "sha512-V/JgOLFCS+R6Vcq0slCuaeWEedNC3ouDUMNIIsacH2VtALiu9mV4LPrHc5cDI8k5aw6J8jwgWWpiTo5RYhmlzvg==",
"license": "ISC",
"dependencies": {
  "minipass": "^3.0.0"
}
},
"engines": {
  "node": ">= 8"
}
},
"node_modules/fs-minipass/node_modules/minipass": {
"version": "3.3.6",
"resolved": "https://registry.npmjs.org/minipass/-/minipass-3.3.6.tgz",
"integrity": "sha512-DxiNIdxSEK+tHG6zOIklvNOWm3hvCrbUrdtzY74U6HKTJxvIDfOUL5W5P2Ghd3DTkhhKPYGqeNUIh5qcM4YBfw==",
"license": "ISC",
```

```
"dependencies": {
  "yallist": "^4.0.0"
},
"engines": {
  "node": ">=8"
}
},
"node_modules/fs.realpath": {
  "version": "1.0.0",
  "resolved": "https://registry.npmjs.org/fs.realpath/-/fs.realpath-1.0.0.tgz",
  "integrity": "sha512-OO0P2H2K6a0hZnAdau5ltzHPI6pUlvI7jMVnxUQRtw4owF2wk8lOSabtGDCTP4Ggrg2MbGnWO9X8K1t4+fGMDw==",
  "license": "ISC"
},
"node_modules/fsevents": {
  "version": "2.3.3",
  "resolved": "https://registry.npmjs.org/fsevents/-/fsevents-2.3.3.tgz",
  "integrity": "sha512-5xoDfX+fL7faATnagmWPpbFtwh/R77WmMMMqqHGS65C3vvB0YHrgF+B1YmZ3441tMj5n63k0212XNoJwzlhffQw==",
  "dev": true,
  "hasInstallScript": true,
  "license": "MIT",
  "optional": true,
  "os": [
    "darwin"
  ],
  "engines": {
    "node": "^8.16.0 || ^10.6.0 || >=11.0.0"
  }
},
"node_modules/function-bind": {
  "version": "1.1.2",
  "resolved": "https://registry.npmjs.org/function-bind/-/function-bind-1.1.2.tgz",
  "integrity": "sha512-5Xpf9l3vQJjGbdVtZGf3Wg7r0T3zv59PMP0aQ7lMbcVxHjW3vJ45V5ZG91WQVkp47W6a8w8Yf35xwQ==",
  "license": "MIT",
  "funding": {
    "url": "https://github.com/sponsors/ljharb"
  }
},
"node_modules/gauge": {
  "version": "3.0.2",
  "resolved": "https://registry.npmjs.org/gauge/-/gauge-3.0.2.tgz",
  "integrity": "sha512-+5J6MS/5XksCuXq++uFRsnUd7Ovu1XenbeuluNRjXyWjgQbPuFhT14lAvsWfqfAmnwluf1OwMjz39HjflPci0Q==",
  "deprecated": "This package is no longer supported.",
  "license": "ISC",
  "dependencies": {
    "aproba": "^1.0.3 || ^2.0.0",
    "color-support": "^1.1.2",
    "console-control-strings": "^1.0.0",
    "has-unicode": "^2.0.1",
    "object-assign": "^4.1.1",
    "signal-exit": "^3.0.0",
    "string-width": "^4.2.3",
    "strip-ansi": "^6.0.1",
    "wide-align": "^1.1.2"
  },
  "engines": {
    "node": ">=10"
  }
},
"node_modules/get-caller-file": {
  "version": "2.0.5",
  "resolved": "https://registry.npmjs.org/get-caller-file/-/get-caller-file-2.0.5.tgz",
  "integrity": "sha512-VJoZ+83lF+M4/HxNRvOe4K+oUoDQF7CpQFt6K1eAPK2e8ZKj8dCVRzVKxdZVbPyxmdqFma3e7kYwLg==",
  "dev": true,
  "license": "ISC",
  "engines": {
    "node": "6.* || 8.* || >= 10.*"
  }
},
"node_modules/get-intrinsic": {
  "version": "1.2.7",
  "resolved": "https://registry.npmjs.org/get-intrinsic/-/get-intrinsic-1.2.7.tgz",
  "integrity": "sha512-VW6Pxhsk0KAOqs3WEed0klDiF/+V7gQOPAvY1jVU/LHmaD/kQO4523aiJuikX/QAKYiW6x8Jh+RJej1almdtCA==",
  "license": "MIT",
  "dependencies": {
    "call-bind-apply-helpers": "^1.0.1",

```

```

    "es-define-property": "^1.0.1",
    "es-errors": "^1.3.0",
    "es-object-atoms": "^1.0.0",
    "function-bind": "^1.1.2",
    "get-proto": "^1.0.0",
    "gopd": "^1.2.0",
    "has-symbols": "^1.1.0",
    "hasown": "^2.0.2",
    "math-intrinsics": "^1.1.0"
  },
  "engines": {
    "node": ">= 0.4"
  },
  "funding": {
    "url": "https://github.com/sponsors/ljharb"
  }
},
"node_modules/get-proto": {
  "version": "1.0.1",
  "resolved": "https://registry.npmjs.org/get-proto/-/get-proto-1.0.1.tgz",
  "integrity": "sha512-sTSfBjoxBp89JvKllefqw7U2CCebSc74kiY6awiGogKtoSGbgjYE/G/+l9sF3MWFPNc9lcoOC4ODfKHfXfmp0g==",
  "license": "MIT",
  "dependencies": {
    "dunder-proto": "^1.0.1",
    "es-object-atoms": "^1.0.0"
  },
  "engines": {
    "node": ">= 0.4"
  }
},
"node_modules/glob": {
  "version": "7.2.3",
  "resolved": "https://registry.npmjs.org/glob/-/glob-7.2.3.tgz",
  "integrity": "sha512-nFR0zLpU2YCaRxwoCJvL6UvCH2JFyFVlVwTLsf21AuHlMskA1hhTdk+LLYJtOlYt9v6dvszD2BGRqBL+iQK9Q==",
  "deprecated": "Glob versions prior to v9 are no longer supported",
  "license": "ISC",
  "dependencies": {
    "fs.realpath": "^1.0.0",
    "inflight": "^1.0.4",
    "inherits": "2",
    "minimatch": "^3.1.1",
    "once": "^1.3.0",
    "path-is-absolute": "^1.0.0"
  },
  "engines": {
    "node": "*"
  },
  "funding": {
    "url": "https://github.com/sponsors/isaacs"
  }
},
"node_modules/glob-parent": {
  "version": "5.1.2",
  "resolved": "https://registry.npmjs.org/glob-parent/-/glob-parent-5.1.2.tgz",
  "integrity": "sha512-AOIgSQcepijYwP3ARnGx+5VnTu2HBYdzGP45eLw1vr3zB3vZLeyed1sC9hnbcOc9/SrMyM5RPQrkGz4aS9Zow==",
  "dev": true,
  "license": "ISC",
  "dependencies": {
    "is-glob": "^4.0.1"
  },
  "engines": {
    "node": ">= 6"
  }
},
"node_modules/gopd": {
  "version": "1.2.0",
  "resolved": "https://registry.npmjs.org/gopd/-/gopd-1.2.0.tgz",
  "integrity": "sha512-1WnK/Y0PjM8LW0Bj8b6zr4wBQb03tHm11bFJ73CZ0eQnE1GfAPV3W/o/17ES88nmKqQMq18hEi2+NRg==",
  "license": "MIT",
  "engines": {
    "node": ">= 0.4"
  },
  "funding": {
    "url": "https://github.com/sponsors/ljharb"
  }
},
"node_modules/has-flag": {
  "version": "3.0.0",

```

```
"resolved": "https://registry.npmjs.org/has-flag/-/has-flag-3.0.0.tgz",
"integrity": "sha512-sKJf1+ceQBr4SMkvQnBDNDtf4TXpVhVGateu0t918bl30FnEb2m4vNLX+VWe/dpjlB+HugGYzW7uQXH98HPEYw==",
"dev": true,
"license": "MIT",
"engines": {
  "node": ">=4"
}
},
"node_modules/has-symbols": {
  "version": "1.1.0",
  "resolved": "https://registry.npmjs.org/has-symbols/-/has-symbols-1.1.0.tgz",
  "integrity": "sha512-1cDNdwJ2Jaohmb3sg4OmKaMBwuC48sYni5HUw2DvsC8LjGTLK9h+eb1X6RyuOHe4hT0ULCW68iomhjUoKUqlPQ==",
  "license": "MIT",
  "engines": {
    "node": ">= 0.4"
  },
  "funding": {
    "url": "https://github.com/sponsors/ljharb"
  }
},
"node_modules/has-unicode": {
  "version": "2.0.1",
  "resolved": "https://registry.npmjs.org/has-unicode/-/has-unicode-2.0.1.tgz",
  "integrity": "sha512-88fG5C29Dy6Z+TRzElk4t4C59n0TSe1c5Yn6PGKiAqbDfAdeCDt65ay4G60M5YWz7I9pw3f5JKJfXUw==",
  "license": "ISC"
},
"node_modules/hasown": {
  "version": "2.0.2",
  "resolved": "https://registry.npmjs.org/hasown/-/hasown-2.0.2.tgz",
  "integrity": "sha512-88fG5C29Dy6Z+TRzElk4t4C59n0TSe1c5Yn6PGKiAqbDfAdeCDt65ay4G60M5YWz7I9pw3f5JKJfXUw==",
  "license": "MIT",
  "dependencies": {
    "function-bind": "^1.1.2"
  },
  "engines": {
    "node": ">= 0.4"
  }
},
"node_modules/he": {
  "version": "1.2.0",
  "resolved": "https://registry.npmjs.org/he/-/he-1.2.0.tgz",
  "integrity": "sha512-F/1DnUGPopORZi0ni+CvrCgHQ5FyEAHRLSApuYWMmrSwoN2Mn/7k+Gl38gJnR7yyDZk6WLXwiGod1JOWNDKGw==",
  "dev": true,
  "license": "MIT",
  "bin": {
    "he": "bin/he"
  }
},
"node_modules/http-errors": {
  "version": "2.0.0",
  "resolved": "https://registry.npmjs.org/http-errors/-/http-errors-2.0.0.tgz",
  "integrity": "sha512-FtwrG/euBzaEjYeRqOgly7G0qviiXoJWnvEH2Z1pIBdXgbyjv34pHTSb9zoeHMyDy33+DWy5Wt9Wo+TURtOYSQ==",
  "license": "MIT",
  "dependencies": {
    "depd": "2.0.0",
    "inherits": "2.0.4",
    "setprototypeof": "1.2.0",
    "statuses": "2.0.1",
    "toidentifier": "1.0.1"
  },
  "engines": {
    "node": ">= 0.8"
  }
},
"node_modules/https-proxy-agent": {
  "version": "5.0.1",
  "resolved": "https://registry.npmjs.org/https-proxy-agent/-/https-proxy-agent-5.0.1.tgz",
  "integrity": "sha512-F1DnUGPopORZi0ni+CvrCgHQ5FyEAHRLSApuYWMmrSwoN2Mn/7k+Gl38gJnR7yyDZk6WLXwiGod1JOWNDKGw==",
  "license": "MIT",
  "dependencies": {
    "agent-base": "6",
    "debug": "4"
  },
  "engines": {
```

```

    "node": ">= 6"
  },
  "node_modules/https-proxy-agent/node_modules/debug": {
    "version": "4.4.0",
    "resolved": "https://registry.npmjs.org/debug/-/debug-4.4.0.tgz",
    "integrity": "sha512-6WTZ/xcCY/T6BALoZHaE4ctp9xm+Z5kY/pzYaCHRFeyVhojxlrM+46y68HA6hr0TcwEssoxNiDEUJQjfPZ/RYA==",
    "license": "MIT",
    "dependencies": {
      "ms": "^2.1.3"
    },
    "engines": {
      "node": ">=6.0"
    },
    "peerDependenciesMeta": {
      "supports-color": {
        "optional": true
      }
    }
  },
  "node_modules/https-proxy-agent/node_modules/ms": {
    "version": "2.1.3",
    "resolved": "https://registry.npmjs.org/ms/-/ms-2.1.3.tgz",
    "integrity": "sha512-6FlzubTLZG3J2a/NVCAleEhJzq5oxgHyaCU9yYXvcLsvoVaHJq/s5xXI6/XXP6tz7R9xAOtHnSO/tXtF3WRTIA==",
    "license": "MIT"
  },
  "node_modules/iconv-lite": {
    "version": "0.4.24",
    "resolved": "https://registry.npmjs.org/iconv-lite/-/iconv-lite-0.4.24.tgz",
    "integrity": "sha512-6v37eG390A39Nh08862X3H2jKcXqIT8V2mB40B/cX0K937d8CB8aL4Y8Vnkb/tUkR243Hr32wZK9Gug==",
    "license": "MIT",
    "dependencies": {
      "safer-buffer": ">= 2.1.2 < 3"
    },
    "engines": {
      "node": ">=0.10.0"
    }
  },
  "node_modules/ignore-by-default": {
    "version": "1.0.1",
    "resolved": "https://registry.npmjs.org/ignore-by-default/-/ignore-by-default-1.0.1.tgz",
    "integrity": "sha512-Iq7A72x14zYImK85dNbBp88W6r/c18v2UR2W9gVtSgZEdHb41JrR4U0Wf8H2sV2jPQcL2XqgA8dWw==",
    "dev": true,
    "license": "ISC"
  },
  "node_modules/inflight": {
    "version": "1.0.6",
    "resolved": "https://registry.npmjs.org/inflight/-/inflight-1.0.6.tgz",
    "integrity": "sha512-k92I/b08q4wwFscXCLvqfsHCjrF7yiXsQuIvVvE7N82W3+aqpuUdBbhfWcy/FZR3/4IgfMgKLOsvPDrGCJA==",
    "deprecated": "This module is not supported, and leaks memory. Do not use it. Check out lru-cache if you want a good and tested way to coalesce async requests by a key value, which is much more comprehensive and powerful.",
    "license": "ISC",
    "dependencies": {
      "once": "^1.3.0",
      "wrappy": "1"
    }
  },
  "node_modules/inherits": {
    "version": "2.0.4",
    "resolved": "https://registry.npmjs.org/inherits/-/inherits-2.0.4.tgz",
    "integrity": "sha512-k/vGaX4/Yla3WzyMCvTQOXYeIhVqOKtnqBduzTHpzpQZzAskKMhZ2K+EnBiSM9zGSoIFeMpXKxa4dYeZIQqewQ==",
    "license": "ISC"
  },
  "node_modules/ipaddr.js": {
    "version": "1.9.1",
    "resolved": "https://registry.npmjs.org/ipaddr.js/-/ipaddr.js-1.9.1.tgz",
    "integrity": "sha512-0KI/60XSToH7GjN1FfSbLoU0+btTicjsQSWQlh/hZyKN8KpmMf7uYwPW3R+akZ6R/w18ZLXSHBYXiYUPO3g==",
    "license": "MIT",
    "engines": {
      "node": ">= 0.10"
    }
  },
  "node_modules/is-binary-path": {
    "version": "2.1.0",
    "resolved": "https://registry.npmjs.org/is-binary-path/-/is-binary-path-2.1.0.tgz",

```

```
"integrity": "sha512-
ZMERYes6pDydyuGidse7OsHxtbI7WVeUEozgR/g7rd0xUimYNIvZRE/K2MgZTjWY725IfelLeVcEM97mmtRGXw==",
  "dev": true,
  "license": "MIT",
  "dependencies": {
    "binary-extensions": "^2.0.0"
  },
  "engines": {
    "node": ">=8"
  }
},
"node_modules/is-extglob": {
  "version": "2.1.1",
  "resolved": "https://registry.npmjs.org/is-extglob/-/is-extglob-2.1.1.tgz",
  "integrity": "sha512-
SbKbANkN603Vi4jEZv49LeVJMn4yGwsbzZworEoyEiutsN3nJYdbO36zfH6J6QEDpOZIFkDtnq5JRxmvl3jsoQ==",
  "dev": true,
  "license": "MIT",
  "engines": {
    "node": ">=0.10.0"
  }
},
"node_modules/is-fullwidth-code-point": {
  "version": "3.0.0",
  "resolved": "https://registry.npmjs.org/is-fullwidth-code-point/-/is-fullwidth-code-point-3.0.0.tgz",
  "integrity": "sha512-
zymm5+u+sCsSWyD9qNaejV3DFVhCKclKdizYaJUuHA83RLjb7nSuGnddCHGv0hk+KY7BMAlsWeK4Ueg6EV6XQg==",
  "license": "MIT",
  "engines": {
    "node": ">=8"
  }
},
"node_modules/is-glob": {
  "version": "4.0.3",
  "resolved": "https://registry.npmjs.org/is-glob/-/is-glob-4.0.3.tgz",
  "integrity": "sha512-
xelSayHH36ZgE7ZWWhi7pW34hNbNl8Ojv5KVmkJD4hBdD3th8Tfk9vYasLM+mXWOZhFkgZfxhLSnrwRr4elSSg==",
  "dev": true,
  "license": "MIT",
  "dependencies": {
    "is-extglob": "^2.1.1"
  },
  "engines": {
    "node": ">=0.10.0"
  }
},
"node_modules/is-number": {
  "version": "7.0.0",
  "resolved": "https://registry.npmjs.org/is-number/-/is-number-7.0.0.tgz",
  "integrity": "sha512-
41Cifkg6e8TylSpdtTpeLVMqvSBEVzTttHvERD741+pnZ8ANv0004MRL43QKPDik9cGvNp6NZWZUBlbGXYxxng==",
  "dev": true,
  "license": "MIT",
  "engines": {
    "node": ">=0.12.0"
  }
},
"node_modules/is-plain-obj": {
  "version": "2.1.0",
  "resolved": "https://registry.npmjs.org/is-plain-obj/-/is-plain-obj-2.1.0.tgz",
  "integrity": "sha512-
YWnfyRwxL/+SsrWYfOpUtz5b3YD+nyfkHvjbcanzk8zgyO4ASD67uVMRt8k5bM4lLMDnXfriRhOpemw+NfT1eA==",
  "dev": true,
  "license": "MIT",
  "engines": {
    "node": ">=8"
  }
},
"node_modules/is-unicode-supported": {
  "version": "0.1.0",
  "resolved": "https://registry.npmjs.org/is-unicode-supported/-/is-unicode-supported-0.1.0.tgz",
  "integrity": "sha512-
knxG2q4UC3u8stRGyAVJCOdxFmv5DZiRcdllaAQXAbSfJya+OhopNotLQrstBhququ4ZpuKbDc/8S6mgXgPFPw==",
  "dev": true,
  "license": "MIT",
  "engines": {
    "node": ">=10"
  },
  "funding": {
    "url": "https://github.com/sponsors/sindresorhus"
  }
}
```

```

},
"node_modules/isexe": {
  "version": "2.0.0",
  "resolved": "https://registry.npmjs.org/isexe/-/isexe-2.0.0.tgz",
  "integrity": "sha512-
RHxMLp9lnKHGHRng9QFhRCMbYAcVpn69smSGcq3f36xjgVWVThj4qqLbTLlq7Ssj8B+flQ1EuCEGI2lKsyQelw==",
  "dev": true,
  "license": "ISC"
},
"node_modules/jackspeak": {
  "version": "3.4.3",
  "resolved": "https://registry.npmjs.org/jackspeak/-/jackspeak-3.4.3.tgz",
  "integrity": "sha512-
OGLzQpz2yfahA/Rd1Y8Cd9SIEsqvXkLVoS/cgwhnhFMDbsQFeZYoj7blZBS9BcamUW96asq/npPWugM+RQBw==",
  "dev": true,
  "license": "BlueOak-1.0.0",
  "dependencies": {
    "@isaacs/cliui": "^8.0.2"
  },
  "funding": {
    "url": "https://github.com/sponsors/isaacs"
  },
  "optionalDependencies": {
    "@pkgjs/parseargs": "^0.11.0"
  },
  "node_modules/js-yaml": {
  "version": "4.1.0",
  "resolved": "https://registry.npmjs.org/js-yaml/-/js-yaml-4.1.0.tgz",
  "integrity": "sha512-
wpxZs9NoxZaJESJGIZTyDEaYpl0FKSA+FB9aJiyemKhMwKxQg63h4T1KJgUGHpTqPDNRcmmYLugrRjLbTvwRA==",
  "dev": true,
  "license": "MIT",
  "dependencies": {
    "argparse": "2.0.1"
  },
  "bin": {
    "js-yaml": "bin/js-yaml.js"
  },
  "node_modules/jsonwebtoken": {
  "version": "9.0.2",
  "resolved": "https://registry.npmjs.org/jsonwebtoken/-/jsonwebtoken-9.0.2.tgz",
  "integrity": "sha512-
PRp66vJ865SSqOlqS8hujT5U4AOgMfhrwYlulhfKaoSCZcirmASQr8CX7cUg+RMih+hgznrjp99o+W4pJLHQ==",
  "license": "MIT",
  "dependencies": {
    "jws": "3.2.2",
    "lodash.includes": "4.3.0",
    "lodash.isboolean": "3.0.3",
    "lodash.isinteger": "4.0.4",
    "lodash.isnumber": "3.0.3",
    "lodash.isplainobject": "4.0.6",
    "lodash.isstring": "4.0.1",
    "lodash.once": "4.0.0",
    "ms": "2.1.1",
    "semver": "7.5.4"
  },
  "engines": {
    "node": ">=12",
    "npm": ">=6"
  },
  "node_modules/jsonwebtoken/node_modules/ms": {
  "version": "2.1.3",
  "resolved": "https://registry.npmjs.org/ms/-/ms-2.1.3.tgz",
  "integrity": "sha512-
cYofQu2Xpom82S6qD778jBDpwwvy39s1L/hrYij2u9AMdQcGRpaBu6kY4mVhuno5kVvi1DAz4aiphA2W11/OAw==",
  "dev": true,
  "license": "MIT"
},
"node_modules/jwa": {
  "version": "1.4.1",
  "resolved": "https://registry.npmjs.org/jwa/-/jwa-1.4.1.tgz",
  "integrity": "sha512-
cYofQu2Xpom82S6qD778jBDpwwvy39s1L/hrYij2u9AMdQcGRpaBu6kY4mVhuno5kVvi1DAz4aiphA2W11/OAw==",
  "dev": true,
  "license": "MIT"
}

```

```
"license": "MIT",
"dependencies": {
  "buffer-equal-constant-time": "1.0.1",
  "ecdsa-sig-formatter": "1.0.11",
  "safe-buffer": "^5.0.1"
}
},
"node_modules/jws": {
  "version": "3.2.2",
  "resolved": "https://registry.npmjs.org/jws/-/jws-3.2.2.tgz",
  "integrity": "sha512-88eF88tJ868p6796652n3B5CnBtu5OgV1t1PzYUx0Hwz9u1/Nj6/90H269NpTqqr66y8Wz5X0j0Z0Q==",
  "license": "MIT",
  "dependencies": {
    "jwa": "^1.4.1",
    "safe-buffer": "^5.0.1"
  }
},
"node_modules/kareem": {
  "version": "2.6.3",
  "resolved": "https://registry.npmjs.org/kareem/-/kareem-2.6.3.tgz",
  "integrity": "sha512-39D880Ew9a0Z35nGMS369O8878UabR0HlL7OgPjK9pPl7qKT0nCs2oA04CYzMnRwXRwvJ3ouyVMS4Nw==",
  "license": "Apache-2.0",
  "engines": {
    "node": ">=12.0.0"
  }
},
"node_modules/locate-path": {
  "version": "6.0.0",
  "resolved": "https://registry.npmjs.org/locate-path/-/locate-path-6.0.0.tgz",
  "integrity": "sha512-iPZK6eYfbxRu3uB4/WZ3EsEIMJFMqAoopl3R+zuq0UjcAm/MO6KCweDgPfP3elTztoKP3KtnVHxTn2NHBSDVUw==",
  "dev": true,
  "license": "MIT",
  "dependencies": {
    "p-locate": "^5.0.0"
  },
  "engines": {
    "node": ">=10"
  },
  "funding": {
    "url": "https://github.com/sponsors/sindresorhus"
  }
},
"node_modules/lodash.get": {
  "version": "4.4.2",
  "resolved": "https://registry.npmjs.org/lodash.get/-/lodash.get-4.4.2.tgz",
  "integrity": "sha512-z+Uw/vLuy6gQe8cfaFWD7p0wVv8fJl3mbzXh33RS+0oW2wvUqiRXiQ69gLWSLpgB5/6sU+r6BlQR0MBILadqTQ==",
  "dev": true,
  "license": "MIT",
  "deprecated": "This package is deprecated. Use the optional chaining (?.) operator instead.",
  "node_modules/lodash.includes": {
    "version": "4.3.0",
    "resolved": "https://registry.npmjs.org/lodash.includes/-/lodash.includes-4.3.0.tgz",
    "integrity": "sha512-W3Bx6mdkRTGtUISOvVD/lbqjTlPPUDTMnIXZFnVwi9NKJ6tiAk6LVdlhZMm17VZisqhKcgzpO5Wz91Pct5b0w==",
    "license": "MIT",
    "node_modules/lodash.isboolean": {
      "version": "3.0.3",
      "resolved": "https://registry.npmjs.org/lodash.isboolean/-/lodash.isboolean-3.0.3.tgz",
      "integrity": "sha512-Bz5mupy2SVbPHURB98VAcw+aHh4vRV5IPNhILUCsOzRmsTmSQ17JluqopAentWoehktxGd9e/hbIXq980/1QJg==",
      "license": "MIT",
      "node_modules/lodash.isinteger": {
        "version": "4.0.4",
        "resolved": "https://registry.npmjs.org/lodash.isinteger/-/lodash.isinteger-4.0.4.tgz",
        "integrity": "sha512-DBwtEWN2caHQ9/imiNeEA5ys1JoRtRfY3d7V9wqtbycnAmTvRRmbHKDV4a0EYc678/dia0jrte4tjYwVBaZUA==",
        "license": "MIT",
        "node_modules/lodash.isnumber": {
          "version": "3.0.3",
          "resolved": "https://registry.npmjs.org/lodash.isnumber/-/lodash.isnumber-3.0.3.tgz",
          "integrity": "sha512-QYqzpfwO3/CWf3XP+Z+tkQsfaLL/EnUlXWVkl5FUPc4sBdTehEqZONuyRt2P67PXAk+NXmTBcc97zw9t1FQrw==",
          "license": "MIT"
        }
      }
    }
  }
}
```

```
  },
  "node_modules/lodash.isplainobject": {
    "version": "4.0.6",
    "resolved": "https://registry.npmjs.org/lodash.isplainobject/-/lodash.isplainobject-4.0.6.tgz",
    "integrity": "sha512-oSXzaWypCMHkPC3NvBEaPHf0KsA5mvPrOPgQWDSbg8n7orZ290M0BmC/jgRZ4vcJ6DTAhjrsSYgdsW/F+MFOBA==",
    "license": "MIT"
  },
  "node_modules/lodash.isstring": {
    "version": "4.0.1",
    "resolved": "https://registry.npmjs.org/lodash.isstring/-/lodash.isstring-4.0.1.tgz",
    "integrity": "sha512-0WJxfxH1wgO3GrbuP+dTTk7op+6L41QCXbGINEmD+ny/G/eCqGzxyCsh7159S+mgDDcoamBw6PC1PS5+wUGgw==",
    "license": "MIT"
  },
  "node_modules/lodash.once": {
    "version": "4.1.1",
    "resolved": "https://registry.npmjs.org/lodash.once/-/lodash.once-4.1.1.tgz",
    "integrity": "sha512-Sb487aTOCr9drQVL8plxOzVhafOjZN9UU54hiN8PU3uAisV7lx1yYNpbNmex2PK6dSJoNTSJUUswT651yww3Mg==",
    "license": "MIT"
  },
  "node_modules/log-symbols": {
    "version": "4.1.0",
    "resolved": "https://registry.npmjs.org/log-symbols/-/log-symbols-4.1.0.tgz",
    "integrity": "sha512-1w4Ca0nB9F5E32j6Zd0y2B4Cu4P8Ks09FV0+Q2o2A7DqoYVpFQIU8j8ZJ7mIm9kL3z988cGqXhXESQ==",
    "dev": true,
    "license": "MIT",
    "dependencies": {
      "chalk": "^4.1.0",
      "is-unicode-supported": "^0.1.0"
    },
    "engines": {
      "node": ">=10"
    },
    "funding": {
      "url": "https://github.com/sponsors/sindresorhus"
    }
  },
  "node_modules/loupe": {
    "version": "3.1.3",
    "resolved": "https://registry.npmjs.org/loupe/-/loupe-3.1.3.tgz",
    "integrity": "sha512-kklp7XSkP78XJEsSxW3712C6teJVoeHHwgo9zJ380de7IYyJ2ISlxojcH2pC5OFLewESmnRi/+XCDIEEVyoug==",
    "dev": true,
    "license": "MIT"
  },
  "node_modules/lru-cache": {
    "version": "10.4.3",
    "resolved": "https://registry.npmjs.org/lru-cache/-/lru-cache-10.4.3.tgz",
    "integrity": "sha512-JNAzZcXrCt42VGLuYz0zfAzDfAvJWW6AfYIDBQyDV5DCll2m5sAmK+OI07s59XfsRsWHP02jAJrRadPRGTt6SQ==",
    "dev": true,
    "license": "ISC"
  },
  "node_modules/make-dir": {
    "version": "3.1.0",
    "resolved": "https://registry.npmjs.org/make-dir/-/make-dir-3.1.0.tgz",
    "integrity": "sha512-0369SgUa839N5l121s9H8UkYHA0IgcYi5lUyScOuDlStE9uDOrXuT8di5H5pn9pcVrQ82cX27rdgLKw==",
    "license": "MIT",
    "dependencies": {
      "semver": "^6.0.0"
    },
    "engines": {
      "node": ">=8"
    },
    "funding": {
      "url": "https://github.com/sponsors/sindresorhus"
    }
  },
  "node_modules/make-dir/node_modules/semver": {
    "version": "6.3.1",
    "resolved": "https://registry.npmjs.org/semver/-/semver-6.3.1.tgz",
    "integrity": "sha512-SWdL88oQhUd0IwQvzZtTl9qmGEKZ/r97/yD0J8Pzo+7W5qg34oMnRG+M6XKFch1UyLF7HCz7Utt1r5Q==",
    "license": "ISC",
    "bin": {
      "semver": "bin/semver.js"
    }
  }
}
```

```
,
"node_modules/math-intrinsics": {
  "version": "1.1.0",
  "resolved": "https://registry.npmjs.org/math-intrinsics/-/math-intrinsics-1.1.0.tgz",
  "integrity": "sha512-
/IxtbwEk5HTPyEwyKX6hGkYXxM9nbj64B+ilVJnC/R6B0pH5G4V3b0pVbL7DBj4tkhBAppQUlf6F6XL9LHu1g==",
  "license": "MIT",
  "engines": {
    "node": ">= 0.4"
  }
},
"node_modules/media-typer": {
  "version": "0.3.0",
  "resolved": "https://registry.npmjs.org/media-typer/-/media-typer-0.3.0.tgz",
  "integrity": "sha512-
dq+qelQ9akHpcOU/gUVRTxVIOkAJ1wR3QAvb4RsVjS8oVoFjDGTc679wJYmUmknUF5HwMLogb5O+a3KxfWapPQ==",
  "license": "MIT",
  "engines": {
    "node": ">= 0.6"
  }
},
"node_modules/memory-pager": {
  "version": "1.5.0",
  "resolved": "https://registry.npmjs.org/memory-pager/-/memory-pager-1.5.0.tgz",
  "integrity": "sha512-
ZS4Bp4r/Zoeq6+NLjpP+0Zzm0pR8whtGPf1XEXKLJBACzGMnS3lt14OinCStjQJm6NU1okjQGSxgEZN8eBYKg==",
  "license": "MIT"
},
"node_modules/merge-descriptors": {
  "version": "1.0.3",
  "resolved": "https://registry.npmjs.org/merge-descriptors/-/merge-descriptors-1.0.3.tgz",
  "integrity": "sha512-
gaNvAS7TZ897/rVaZ0nMtAyxNyi/pdbjbAwUpFQpN70GqnVfOiXpeUUMKRBmzXaSQ8DdTX4/0ms62r2K+hE6mQ==",
  "license": "MIT",
  "funding": {
    "url": "https://github.com/sponsors/sindresorhus"
  }
},
"node_modules/methods": {
  "version": "1.1.2",
  "resolved": "https://registry.npmjs.org/methods/-/methods-1.1.2.tgz",
  "integrity": "sha512-
iclAHeNqNm68zFtnZ0e+1L2yUldvzNoauKU4WBA3VvH/vPFieF7qfRlwUZU+DA9P9bPXIS90ulxoUoCH23sV2w==",
  "license": "MIT",
  "engines": {
    "node": ">= 0.6"
  }
},
"node_modules/mime": {
  "version": "1.6.0",
  "resolved": "https://registry.npmjs.org/mime/-/mime-1.6.0.tgz",
  "integrity": "sha512-
x0Vn8spl+wuJ1O6S7gnbaQg8Pxh4NNHb7KSINmEWKiPE4RKOpIvijn+NkmYmmRgP68mc70j2EbeTFRsrswaQeg==",
  "license": "MIT",
  "bin": {
    "mime": "cli.js"
  },
  "engines": {
    "node": ">=4"
  }
},
"node_modules/mime-db": {
  "version": "1.52.0",
  "resolved": "https://registry.npmjs.org/mime-db/-/mime-db-1.52.0.tgz",
  "integrity": "sha512-
sPU4uV7dYlvtWJxwwwHD0PuihVNIe7TyAbQ5SWxDCB9mUYvOgroQOwYQQOKPJ8ClbE+1ETVlOoK1UC2nU3gYvg==",
  "license": "MIT",
  "engines": {
    "node": ">= 0.6"
  }
},
"node_modules/mime-types": {
  "version": "2.1.35",
  "resolved": "https://registry.npmjs.org/mime-types/-/mime-types-2.1.35.tgz",
  "integrity": "sha512-
ZDY+bPm5zTTF+YpCrAU9nK0UglCYPT0QtT1NZWFv4s++TNkcgVaT0g6+4R2u14MjJqzysHB1zXuWL50hzaeXiw==",
  "license": "MIT",
  "dependencies": {
    "mime-db": "1.52.0"
  },
  "engines": {
```

```
    "node": ">= 0.6"
  }
},
"node_modules/minimatch": {
  "version": "3.1.2",
  "resolved": "https://registry.npmjs.org/minimatch/-/minimatch-3.1.2.tgz",
  "integrity": "sha512-17p63hRiAjw1NDEww1W7i37+BylR0W05XQQAzZ3VOcL0PNybwpmfV/N05zFAzwQ9USyEcX6t3UO+K5aqBQOIHw==",
  "license": "ISC",
  "dependencies": {
    "brace-expansion": "^1.1.7"
  },
  "engines": {
    "node": "*"
  }
},
"node_modules/minipass": {
  "version": "5.0.0",
  "resolved": "https://registry.npmjs.org/minipass/-/minipass-5.0.0.tgz",
  "integrity": "sha512-3FnJYuehv9k6ovOEbvOswadCDPX1piCfhV8ncmYtHOjuPwylVWsgHTLo7rabjC3Rx5xD4HDx8Wm1xnMF7S5qFQ==",
  "license": "ISC",
  "engines": {
    "node": ">=8"
  }
},
"node_modules/minizlib": {
  "version": "2.1.2",
  "resolved": "https://registry.npmjs.org/minizlib/-/minizlib-2.1.2.tgz",
  "integrity": "sha512-bAxsR8BVfj60DWXHE3u30oHzfl4G7khkSuPW+qvpd7jFRHm7dLxOjUk1EHACJ/hxLY8phGJ0YhYHZo7jil7Qdg==",
  "license": "MIT",
  "dependencies": {
    "minipass": "^3.0.0",
    "yallist": "^4.0.0"
  },
  "engines": {
    "node": ">= 8"
  }
},
"node_modules/minizlib/node_modules/minipass": {
  "version": "3.3.6",
  "resolved": "https://registry.npmjs.org/minipass/-/minipass-3.3.6.tgz",
  "integrity": "sha512-DxiNidxSEK+tHG6zOIklvNOwm3hvCrbUrdtzY74U6HKTJxvIDfOUL5W5P2Ghd3DTkhhKPYGqeNUIh5qcM4YBfw==",
  "license": "ISC",
  "dependencies": {
    "yallist": "^4.0.0"
  },
  "engines": {
    "node": ">=8"
  }
},
"node_modules/mkdirp": {
  "version": "1.0.4",
  "resolved": "https://registry.npmjs.org/mkdirp/-/mkdirp-1.0.4.tgz",
  "integrity": "sha512-vVqVZyqf3WLx2Shd0qJ9xuvqgAyKPLAiqITetqW0oIUjzo3PePDd6fW9Ifz30ef7Ysp/oiWqbhszeGWW2T6Gzw==",
  "license": "MIT",
  "bin": {
    "mkdirp": "bin/cmd.js"
  },
  "engines": {
    "node": ">=10"
  }
},
"node_modules/mocha": {
  "version": "11.1.0",
  "resolved": "https://registry.npmjs.org/mocha/-/mocha-11.1.0.tgz",
  "integrity": "sha512-8uJRS5RTC2NgpY3GrYcgpZrsEd9zKbPDpob1RezyR2upGHRQtHWofmzTMzTMSV6dru3tj5Ukt0+Vnq1qhFEWAg==",
  "dev": true,
  "license": "MIT",
  "dependencies": {
    "ansi-colors": "^4.1.3",
    "browser-stdout": "^1.3.1",
    "chokidar": "^3.5.3",
    "debug": "^4.3.5",
    "diff": "^5.2.0",
    "escape-string-regexp": "^4.0.0",
    "find-up": "^5.0.0",

```

```

"glob": "^10.4.5",
"he": "^1.2.0",
"js-yaml": "^4.1.0",
"log-symbols": "^4.1.0",
"minimatch": "^5.1.6",
"ms": "^2.1.3",
"serialize-javascript": "^6.0.2",
"strip-json-comments": "^3.1.1",
"supports-color": "^8.1.1",
"workerpool": "^6.5.1",
"yargs": "^17.7.2",
"yargs-parser": "^21.1.1",
"yargs-unparser": "^2.0.0"
},
"bin": {
  "_mocha": "bin/_mocha",
  "mocha": "bin/mocha.js"
},
"engines": {
  "node": "^18.18.0 || ^20.9.0 || >=21.1.0"
}
},
"node_modules/mocha/node_modules/brace-expansion": {
  "version": "2.0.1",
  "resolved": "https://registry.npmjs.org/brace-expansion/-/brace-expansion-2.0.1.tgz",
  "integrity": "sha512-XnAlvQ8eM+kC6aULx6wuQiwVsnzsi9d3WxzV3FpWTGA19F621kwdbAcFKXgKUHZWsy+mY6iL1sHTxWEFCytDA==",
  "dev": true,
  "license": "MIT",
  "dependencies": {
    "balanced-match": "^1.0.0"
  }
},
"node_modules/mocha/node_modules/debug": {
  "version": "4.4.0",
  "resolved": "https://registry.npmjs.org/debug/-/debug-4.4.0.tgz",
  "integrity": "sha512-6WTZ/1xCY/T6BALoZHaE4ctp9xm+Z5kY/pzYaChRFeyVhojxlrn+46y68HA6hr0TcwEssoxNiDEUJQjfPZ/RYA==",
  "dev": true,
  "license": "MIT",
  "dependencies": {
    "ms": "^2.1.3"
  }
},
"engines": {
  "node": ">=6.0"
},
"peerDependenciesMeta": {
  "supports-color": {
    "optional": true
  }
}
},
"node_modules/mocha/node_modules/glob": {
  "version": "10.4.5",
  "resolved": "https://registry.npmjs.org/glob/-/glob-10.4.5.tgz",
  "integrity": "sha512-7BvR0k6xjo7d4A/PxYLBUCfb6c+Vpd2/mB2yRDlew7Jb5hEXiCD9ibfO7wpk8i4sevK6DFny9h7EYbM3/sHg==",
  "dev": true,
  "license": "ISC",
  "dependencies": {
    "foreground-child": "^3.1.0",
    "jackspeak": "^3.1.2",
    "minimatch": "^9.0.4",
    "minipass": "^7.1.2",
    "package-json-from-dist": "^1.0.0",
    "path-scurry": "^1.11.1"
  }
},
"bin": {
  "glob": "dist/esm/bin.mjs"
},
"funding": {
  "url": "https://github.com/sponsors/isaacs"
}
},
"node_modules/mocha/node_modules/glob/node_modules/minimatch": {
  "version": "9.0.5",
  "resolved": "https://registry.npmjs.org/minimatch/-/minimatch-9.0.5.tgz",
  "integrity": "sha512-G6T0Z48xgozx7587koeX9Ys2NYy6Gmv//P89sEte9V9whlApMNF4idKxnW2QtCcLiTWLb/wfCabAtAFWhhBow==",
  "dev": true,
  "license": "ISC",

```

```

"dependencies": {
  "brace-expansion": "^2.0.1"
},
"engines": {
  "node": ">=16 || 14 >=14.17"
},
"funding": {
  "url": "https://github.com/sponsors/isaacs"
}
},
"node_modules/mocha/node_modules/has-flag": {
  "version": "4.0.0",
  "resolved": "https://registry.npmjs.org/has-flag/-/has-flag-4.0.0.tgz",
  "integrity": "sha512-EykJT/Q1KJTWctppgIAgfSO0tKVuZUjhgMr17kqTumMl6Afv3EISleU7qZUzoXDFTAHTDC4NOoG/ZxU3EvlMPQ==",
  "dev": true,
  "license": "MIT",
  "engines": {
    "node": ">=8"
  }
},
"node_modules/mocha/node_modules/minimatch": {
  "version": "5.1.6",
  "resolved": "https://registry.npmjs.org/minimatch/-/minimatch-5.1.6.tgz",
  "integrity": "sha512-lKldVbiZ0LxdL4Pc3GCGPcu8+CGoH29jqScUI6KjQ54gmvW6vYbhjHN2n5DPoW64mZsD29i4M38bHq+2n5yHw==",
  "dev": true,
  "license": "ISC",
  "dependencies": {
    "brace-expansion": "^2.0.1"
  },
  "engines": {
    "node": ">=10"
  }
},
"node_modules/mocha/node_modules/minipass": {
  "version": "7.1.2",
  "resolved": "https://registry.npmjs.org/minipass/-/minipass-7.1.2.tgz",
  "integrity": "sha512-qOOzS1cBTWYF4BH8fVePDBOO9iptMnGUEZwNc/cMWnTV2nVLZ7VoNWEPhkYczZA0pdoA7dl6e7FL659nX9S2aw==",
  "dev": true,
  "license": "ISC",
  "engines": {
    "node": ">=16 || 14 >=14.17"
  }
},
"node_modules/mocha/node_modules/ms": {
  "version": "2.1.3",
  "resolved": "https://registry.npmjs.org/ms/-/ms-2.1.3.tgz",
  "integrity": "sha512-6FbWgF4807S1q4ccX/yAlPFtwZBKlJ9cA/MIqiyZTdAgXg2g8WHhN43LHBE8q+XehdRO/iyYlP1AImPTh9Bz==",
  "dev": true,
  "license": "MIT"
},
"node_modules/mocha/node_modules/supports-color": {
  "version": "8.1.1",
  "resolved": "https://registry.npmjs.org/supports-color/-/supports-color-8.1.1.tgz",
  "integrity": "sha512-i9w3g3B5UgA2E2x2Ox1O4Iq7R3E35UpNcWtArL1gjbYxZr7Pcs2n4BfdYQstjU1OQ2TiuWB3xMQ1ayBA==",
  "dev": true,
  "license": "MIT",
  "dependencies": {
    "has-flag": "^4.0.0"
  },
  "engines": {
    "node": ">=10"
  }
},
"funding": {
  "url": "https://github.com/chalk/supports-color?sponsor=1"
}
},
"node_modules/mongodb": {
  "version": "6.12.0",
  "resolved": "https://registry.npmjs.org/mongodb/-/mongodb-6.12.0.tgz",
  "integrity": "sha512-RM7AHlvYfS7jv7+BXund/kR64DryVI+cHbVAy9P61fmb1RcWZqOW1/Wj2YhqMCx+MuYhqTRGv7AwHBzmsCKBfA==",
  "license": "Apache-2.0",
  "dependencies": {
    "@mongodb-js/saslprep": "^1.1.9",
    "bson": "^6.10.1",
    "mongodb-connection-string-url": "^3.0.0"
  }
},

```

```

"engines": {
  "node": ">=16.20.1"
},
"peerDependencies": {
  "@aws-sdk/credential-providers": "^3.188.0",
  "@mongodb-js/zstd": "^1.1.0 || ^2.0.0",
  "gcp-metadata": "^5.2.0",
  "kerberos": "^2.0.1",
  "mongodb-client-encryption": ">=6.0.0 <7",
  "snappy": "^7.2.2",
  "socks": "^2.7.1"
},
"peerDependenciesMeta": {
  "@aws-sdk/credential-providers": {
    "optional": true
  },
  "@mongodb-js/zstd": {
    "optional": true
  },
  "gcp-metadata": {
    "optional": true
  },
  "kerberos": {
    "optional": true
  },
  "mongodb-client-encryption": {
    "optional": true
  },
  "snappy": {
    "optional": true
  },
  "socks": {
    "optional": true
  }
},
"node_modules/mongodb-connection-string-url": {
  "version": "3.0.2",
  "resolved": "https://registry.npmjs.org/mongodb-connection-string-url/-/mongodb-connection-string-url-3.0.2.tgz",
  "integrity": "sha512-1R0pR+G6p5sgUzoITC1r34kQB2yXlELv36rtaGdgdN/yj1fD1NAkbcW54w64jLomX+sSN7Y7p+h3/eF7upzA==",
  "license": "Apache-2.0",
  "dependencies": {
    "@types/whatwg-url": "^11.0.2",
    "whatwg-url": "^14.1.0 || ^13.0.0"
  }
},
"node_modules/mongoose": {
  "version": "8.9.6",
  "resolved": "https://registry.npmjs.org/mongoose/-/mongoose-8.9.6.tgz",
  "integrity": "sha512-1ipLvXwNPVuuuq5H3InSD0lpARH3DlCoC6emnMVJvweTwXU29uxDJWxMsNpERDQt8JMvYF1HGVuTK+Id2BIQLCA==",
  "license": "MIT",
  "dependencies": {
    "bson": "^6.10.1",
    "kareem": "2.6.3",
    "mongodb": "~6.12.0",
    "mpath": "0.9.0",
    "mquery": "5.0.0",
    "ms": "2.1.3",
    "sift": "17.1.3"
  },
  "engines": {
    "node": ">=16.20.1"
  },
  "funding": {
    "type": "opencollective",
    "url": "https://opencollective.com/mongoose"
  }
},
"node_modules/mongoose/node_modules/ms": {
  "version": "2.1.3",
  "resolved": "https://registry.npmjs.org/ms/-/ms-2.1.3.tgz",
  "integrity": "sha512-6F2WJ2C6IYFspYEm862IquO2v6XFW38qVZ83l8WFbNtCwbdPUDCv1saO0WUup4EFVYsZz60kHgVx5xR5g==",
  "license": "MIT"
},
"node_modules/mpath": {
  "version": "0.9.0",
  "resolved": "https://registry.npmjs.org/mpath/-/mpath-0.9.0.tgz",
  "integrity": "sha512-ROvyK8D8F4pKpdW7csb1Ui4pVq/85B7ZuXbF51P8y8VbWwW6Wd8a941xndBdYU7I8I9DStw72L6UHq==",
  "license": "MIT"
}

```

```
"license": "MIT",
"engines": {
  "node": ">=4.0.0"
}
},
"node_modules/mquery": {
  "version": "5.0.0",
  "resolved": "https://registry.npmjs.org/mquery/-/mquery-5.0.0.tgz",
  "integrity": "sha512-iQMncpmEK8R8ncT8HJGsGc9Dsp8xcgYmVSbs5jgnm1lFHTZqMJTUWTDx1LBO8+mK3tPNZWFLBghQEIOULSTHZg==",
  "license": "MIT",
  "dependencies": {
    "debug": "4.x"
  },
  "engines": {
    "node": ">=14.0.0"
  }
},
"node_modules/mquery/node_modules/debug": {
  "version": "4.4.0",
  "resolved": "https://registry.npmjs.org/debug/-/debug-4.4.0.tgz",
  "integrity": "sha512-6WTZ/lxCY/T6BALoZHaE4ctp9xm+Z5kY/pzYaCHRFeyVhojxlrM+46y68HA6hr0TcwEssoxNiDEUJQjfPZ/RYA==",
  "license": "MIT",
  "dependencies": {
    "ms": "^2.1.3"
  },
  "engines": {
    "node": ">=6.0"
  },
  "peerDependenciesMeta": {
    "supports-color": {
      "optional": true
    }
  }
},
"node_modules/mquery/node_modules/ms": {
  "version": "2.1.3",
  "resolved": "https://registry.npmjs.org/ms/-/ms-2.1.3.tgz",
  "integrity": "sha512-6FlzubTLZG3J2a/NVCAleEhjq5oxgHyaCU9yYXvcLsvoVaHJq/s5xXI6/XXP6tz7R9xAOtHnSO/tXtF3WRTIA==",
  "license": "MIT"
},
"node_modules/ms": {
  "version": "2.0.0",
  "resolved": "https://registry.npmjs.org/ms/-/ms-2.0.0.tgz",
  "integrity": "sha512-Tpp60P6IUJDTuOq/5Z8cdskzJujfwqOTkrwIwj7IRISpnkJnT6SyJ4PCPnGMoFjC9ddhal5KVIYtAt97ix05A==",
  "license": "MIT"
},
"node_modules/negotiator": {
  "version": "0.6.3",
  "resolved": "https://registry.npmjs.org/negotiator/-/negotiator-0.6.3.tgz",
  "integrity": "sha512-6861/8nzA9A3PrX5W4549H+VbfizE2EiM8M8f8z4dWNEk995R7b8W81e45EI7dM127MDYsCn+G5ANZyUw==",
  "license": "MIT",
  "engines": {
    "node": ">= 0.6"
  }
},
"node_modules/nise": {
  "version": "6.1.1",
  "resolved": "https://registry.npmjs.org/nise/-/nise-6.1.1.tgz",
  "integrity": "sha512-aMSAzLVY7LyeM60gvBS423nBmIPP+Wy7St7hsb+8/fc1HmeoHJfLO8CKse4u3BtOZvQLJghYPI2i/1WZrEj5/g==",
  "dev": true,
  "license": "BSD-3-Clause",
  "dependencies": {
    "@sinonjs/commons": "^3.0.1",
    "@sinonjs/fake-timers": "^13.0.1",
    "@sinonjs/text-encoding": "^0.7.3",
    "just-extend": "^6.2.0",
    "path-to-regexp": "^8.1.0"
  }
},
"node_modules/nise/node_modules/path-to-regexp": {
  "version": "8.2.0",
  "resolved": "https://registry.npmjs.org/path-to-regexp/-/path-to-regexp-8.2.0.tgz",
  "integrity": "sha512-TdEz6ZwvUjOw9ZwzPbnJlRnY8vF7PGE+YR8Lyp0E4MP05XJ0ZW4wQBZuX7F544zJ0bfpY4E24UOqV7W2Wg==",
  "dev": true,
  "license": "MIT",
  "engines": {

```

```
    "node": ">=16"
  }
},
"node_modules/node-addon-api": {
  "version": "5.1.0",
  "resolved": "https://registry.npmjs.org/node-addon-api/-/node-addon-api-5.1.0.tgz",
  "integrity": "sha512-e90GgfkEpknoWDq+VY8OyvYhFEzBk6jlYbRKdIlyTiAXIVJ8PyBaKb0rp7oDtoddbdoHWhq8wwr+XZ81F1rpNdA==",
  "license": "MIT"
},
"node_modules/node-fetch": {
  "version": "2.7.0",
  "resolved": "https://registry.npmjs.org/node-fetch/-/node-fetch-2.7.0.tgz",
  "integrity": "sha512-bC5AD7dH06N36n2HC4327ZP6ADf22U1a5qYg7yU1o4a5XU8u8qfVWj38eZL4iVYB7J4XqyGzj83e48==",
  "license": "MIT",
  "dependencies": {
    "whatwg-url": "^5.0.0"
  },
  "engines": {
    "node": "4.x || >=6.0.0"
  },
  "peerDependencies": {
    "encoding": "^0.1.0"
  },
  "peerDependenciesMeta": {
    "encoding": {
      "optional": true
    }
  }
},
"node_modules/node-fetch/node_modules/tr46": {
  "version": "0.0.3",
  "resolved": "https://registry.npmjs.org/tr46/-/tr46-0.0.3.tgz",
  "integrity": "sha512-N3WMsUqV66lT30CrXNbEjx4GEwlow3v6rr4mCcv6prnfwHs01rkgYFdjPNBYd9br7LpXV1+Emh01fHnq2Gdgrw==",
  "license": "MIT"
},
"node_modules/node-fetch/node_modules/webidl-conversions": {
  "version": "3.0.1",
  "resolved": "https://registry.npmjs.org/webidl-conversions/-/webidl-conversions-3.0.1.tgz",
  "integrity": "sha512-2BtWqYXJp9yNoYv6y0+8QJv28Q8aCk9UgO5WbMT4B78wzQW4S0Bd8P7YwQZS8gdyu0HTxmSYEqI5t0Z==",
  "license": "BSD-2-Clause"
},
"node_modules/node-fetch/node_modules/whatwg-url": {
  "version": "5.0.0",
  "resolved": "https://registry.npmjs.org/whatwg-url/-/whatwg-url-5.0.0.tgz",
  "integrity": "sha512-0L3/200tX3wUP87F18Jw42B88ArdI6iGOvxd4UtNro8UqztF2D67zIbBC94I3mYk147C8Qz9HwIvD/A==",
  "license": "MIT",
  "dependencies": {
    "tr46": "~0.0.3",
    "webidl-conversions": "^3.0.0"
  }
},
"node_modules/nodemon": {
  "version": "3.1.9",
  "resolved": "https://registry.npmjs.org/nodemon/-/nodemon-3.1.9.tgz",
  "integrity": "sha512-hdr1olb2p6ZSxu3PB2JWWYS7ZQ0qvaZsc3hK8DR8f02kRzc8rjYmxAlvdz+aYC+8F2IjNaB7HMcSDg8nQpJxyg==",
  "dev": true,
  "license": "MIT",
  "dependencies": {
    "chokidar": "^3.5.2",
    "debug": "^4",
    "ignore-by-default": "^1.0.1",
    "minimatch": "^3.1.2",
    "pstree.remy": "^1.1.8",
    "semver": "^7.5.3",
    "simple-update-notifier": "^2.0.0",
    "supports-color": "^5.5.0",
    "touch": "^3.1.0",
    "undefsafe": "^2.0.5"
  },
  "bin": {
    "nodemon": "bin/nodemon.js"
  },
  "engines": {
    "node": ">=10"
  }
},
```

```
"funding": {
  "type": "opencollective",
  "url": "https://opencollective.com/nodemon"
},
},
"node_modules/nodemon/node_modules/debug": {
  "version": "4.4.0",
  "resolved": "https://registry.npmjs.org/debug/-/debug-4.4.0.tgz",
  "integrity": "sha512-6WTZ/lxCY/T6BALoZHaE4ctp9xm+Z5kY/pzYaCHRFeyVhojxlrn+46y68HA6hr0TcwEssoxNiDEUJQjfPZ/RYA==",
  "dev": true,
  "license": "MIT",
  "dependencies": {
    "ms": "^2.1.3"
  },
  "engines": {
    "node": ">=6.0"
  },
  "peerDependenciesMeta": {
    "supports-color": {
      "optional": true
    }
  }
},
"node_modules/nodemon/node_modules/ms": {
  "version": "2.1.3",
  "resolved": "https://registry.npmjs.org/ms/-/ms-2.1.3.tgz",
  "integrity": "sha512-6FlzubTLZG3J2a/NVCAleEhJzq5oxgHyaCU9yYXvcLsvoVaHJq/s5xXI6/XXP6tz7R9xAOtHnSO/tXtF3WRTIA==",
  "dev": true,
  "license": "MIT"
},
"node_modules/nopt": {
  "version": "5.0.0",
  "resolved": "https://registry.npmjs.org/nopt/-/nopt-5.0.0.tgz",
  "integrity": "sha512-TXk6G1n2tyKtDFrxTj1P269yL99fqvMC58XArb63rvKz6ztm6GqUKz4/VcR1qLpR88OZeQ50E7vz1VwL0h2Y=",
  "license": "ISC",
  "dependencies": {
    "abbrev": "1"
  },
  "bin": {
    "nopt": "bin/nopt.js"
  },
  "engines": {
    "node": ">=6"
  }
},
"node_modules/normalize-path": {
  "version": "3.0.0",
  "resolved": "https://registry.npmjs.org/normalize-path/-/normalize-path-3.0.0.tgz",
  "integrity": "sha512-6eZs5Ls3WtCisHWP9S2GUy8dqkPGi4BVSz3GaQIE6ezub0512ESztXUwUB6C6IKbQkY2Pnb/mD4WYojCRwcvLA==",
  "dev": true,
  "license": "MIT",
  "engines": {
    "node": ">=0.10.0"
  }
},
"node_modules/npmlog": {
  "version": "5.0.1",
  "resolved": "https://registry.npmjs.org/npmlog/-/npmlog-5.0.1.tgz",
  "integrity": "sha512-3E1Jh4rfMR0qj6z65jP2W7VzF8RFZhi4uib49kZEY+Ba6hTGx20qkX8Wg/0A8ImS8oGofBDD+TdOu7e2KRiaA==",
  "deprecated": "This package is no longer supported.",
  "license": "ISC",
  "dependencies": {
    "are-we-there-yet": "^2.0.0",
    "console-control-strings": "^1.1.0",
    "gauge": "^3.0.0",
    "set-blocking": "^2.0.0"
  },
  "engines": {
    "node": ">=0.10.0"
  }
},
"node_modules/object-assign": {
  "version": "4.1.1",
  "resolved": "https://registry.npmjs.org/object-assign/-/object-assign-4.1.1.tgz",
  "integrity": "sha512-jVjwYUjv06hT3SBCDeJfJdtLKBjIw0bmub505Hk0LgYb8m0jNv0RKsFHe9ZClWCAGvD5kYhghZUiQvZ9TA==",
  "license": "MIT",
  "engines": {
    "node": ">=0.10.0"
  }
}
```

```

},
"node_modules/object-inspect": {
  "version": "1.13.3",
  "resolved": "https://registry.npmjs.org/object-inspect/-/object-inspect-1.13.3.tgz",
  "integrity": "sha512-kDCGlboxkDSXE3euJZZXzc6to7fCrKHNI/hSRQnRuQ+BWjFNzZwiFF8fj/6o2t2G9/jTj8PSIYtFCLeLZEeRpA==",
  "license": "MIT",
  "engines": {
    "node": ">= 0.4"
  },
  },
  "funding": {
    "url": "https://github.com/sponsors/ljharb"
  }
},
"node_modules/on-finished": {
  "version": "2.4.1",
  "resolved": "https://registry.npmjs.org/on-finished/-/on-finished-2.4.1.tgz",
  "integrity": "sha512-oVHzU1n96ahKsTucPwq3yKIeK3mJlK3ZL30eYqY3QnH8KtO80+rZnQA8kbPsj23KJk3Oz7oW6ky+8gQ==",
  "license": "MIT",
  "dependencies": {
    "ee-first": "1.1.1"
  },
  "engines": {
    "node": ">= 0.8"
  }
},
"node_modules/once": {
  "version": "1.4.0",
  "resolved": "https://registry.npmjs.org/once/-/once-1.4.0.tgz",
  "integrity": "sha512-lNaJgkl+2Q5URQBBkccEKHTQOPaXdUxnZZEIQTZY0MFUAuaEqe1E+Nyvgdz/alyNi6Z9MzO5dv1H8n58/GELp3+w==",
  "license": "ISC",
  "dependencies": {
    "wrappy": "1"
  }
},
"node_modules/p-limit": {
  "version": "3.1.0",
  "resolved": "https://registry.npmjs.org/p-limit/-/p-limit-3.1.0.tgz",
  "integrity": "sha512-TYOanM3wGwNGsZn2cVTYPArw454xnXj5qmWF1bEoAc4+cU/ol7GVh7odevjp1FNHduHc3KZMcFduxU5Xc6uJRQ==",
  "dev": true,
  "license": "MIT",
  "dependencies": {
    "yocto-queue": "^0.1.0"
  },
  "engines": {
    "node": ">=10"
  }
},
"node_modules/p-locate": {
  "version": "5.0.0",
  "resolved": "https://registry.npmjs.org/p-locate/-/p-locate-5.0.0.tgz",
  "integrity": "sha512-LnLaS6NpTUoVNUaTYjuzwZdOXVnc0aY8oDfYr8os31YBZjN3DQoQ6aU/+89b5UocZs8U+eG1ZwMvV6Y==",
  "dev": true,
  "license": "MIT",
  "dependencies": {
    "p-limit": "^3.0.2"
  },
  "engines": {
    "node": ">=10"
  }
},
"node_modules/package-json-from-dist": {
  "version": "1.0.1",
  "resolved": "https://registry.npmjs.org/package-json-from-dist/-/package-json-from-dist-1.0.1.tgz",
  "integrity": "sha512-UEZIS3/by4OC8vL3P2dTXXREtpEbLi2Ni5vIraD/5UtrkFX/tNbWjTSRAGC/+7CAo2plcBaRgWmcBBHcsaClw==",
  "dev": true,
  "license": "BlueOak-1.0.0"
},
"node_modules/parseurl": {
  "version": "1.3.3",
  "resolved": "https://registry.npmjs.org/parseurl/-/parseurl-1.3.3.tgz",

```

```
"integrity": "sha512-
CiyeOxFT/JZyN5m0z9PfXw4SCBJ6Sygz1Dpl0wqjlhDEGGBP1GnsUVEL0p63hoG1fcj3fHynXi9NYO4nWOL+qQ==",
"license": "MIT",
"engines": {
  "node": ">= 0.8"
}
},
"node_modules/path-exists": {
  "version": "4.0.0",
  "resolved": "https://registry.npmjs.org/path-exists/-/path-exists-4.0.0.tgz",
  "integrity": "sha512-
ak9Qy5Q7jYb2Wwcey5Fpvg2KoAc/ZlhLSLOSbmRmygPsGwkVVt0fZa0qrtMz+m6UJAHfZQ8FnmB4MG4LWY7/w==",
  "dev": true,
  "license": "MIT",
  "engines": {
    "node": ">=8"
  }
},
"node_modules/path-is-absolute": {
  "version": "1.0.1",
  "resolved": "https://registry.npmjs.org/path-is-absolute/-/path-is-absolute-1.0.1.tgz",
  "integrity": "sha512-
AVbw3UJ2e9bq64vSaS9Am0fje1Pa8pbGqTTsmXfaliMpnR5DlDhfJOuLj9Sf95ZPVDAUerDfEk88MPmPe7UCQg==",
  "license": "MIT",
  "engines": {
    "node": ">=0.10.0"
  }
},
"node_modules/path-key": {
  "version": "3.1.1",
  "resolved": "https://registry.npmjs.org/path-key/-/path-key-3.1.1.tgz",
  "integrity": "sha512-
ojmeN0qd+y0jszEtoY48r0Peq5dwMEklCOu6Q5f41lfkswXuKtYrhgoTpLnyIcHm24Uhxq+5Tqm2lnSwLhE6Q==",
  "dev": true,
  "license": "MIT",
  "engines": {
    "node": ">=8"
  }
},
"node_modules/path-scurry": {
  "version": "1.11.1",
  "resolved": "https://registry.npmjs.org/path-scurry/-/path-scurry-1.11.1.tgz",
  "integrity": "sha512-
Xa4Nw17FS9ApQFJ9umLiJS4orGjm7ZzwUrwamcGQuHSzDyth9boKDaycYdDcZDuqYATXw4HFXgaqWTctW/v1HA==",
  "dev": true,
  "license": "BlueOak-1.0.0",
  "dependencies": {
    "lru-cache": "^10.2.0",
    "minipass": "^5.0.0 || ^6.0.2 || ^7.0.0"
  },
  "engines": {
    "node": ">=16 || 14 >=14.18"
  },
  "funding": {
    "url": "https://github.com/sponsors/isaacs"
  }
},
"node_modules/path-to-regexp": {
  "version": "0.1.12",
  "resolved": "https://registry.npmjs.org/path-to-regexp/-/path-to-regexp-0.1.12.tgz",
  "integrity": "sha512-
RA1GjUVMnvYFXuqovrEqZoxXW5NUZqbWkTYz/Tt7nXerk0LbLbLQmrsgdeOxV5SFHf0UDggjS/bSeOZwt1pmEQ==",
  "license": "MIT"
},
"node_modules/pathval": {
  "version": "2.0.0",
  "resolved": "https://registry.npmjs.org/pathval/-/pathval-2.0.0.tgz",
  "integrity": "sha512-
vE7JKRyES09KiunauX7nd2Q9/L7lhok4smP9RZTDeD4MVz72Dp2qNFVz39Nz5a0FVEW0BJR6C0DYrq6unoziZA==",
  "dev": true,
  "license": "MIT",
  "engines": {
    "node": ">= 14.16"
  }
},
"node_modules/picomatch": {
  "version": "2.3.1",
  "resolved": "https://registry.npmjs.org/picomatch/-/picomatch-2.3.1.tgz",
  "integrity": "sha512-
JU3teHTNjmE2VCGFzuY8EXzCDVwEqB2a8fslvwaStHhAWJJeVd1o1QD80CU6+ZdEXXSLbSsuLwJjkCBWqRQUVA==",
  "dev": true,
```

```
"license": "MIT",
"engines": {
  "node": ">=8.6"
},
"funding": {
  "url": "https://github.com/sponsors/jonschlinkert"
}
},
"node_modules/proxy-addr": {
  "version": "2.0.7",
  "resolved": "https://registry.npmjs.org/proxy-addr/-/proxy-addr-2.0.7.tgz",
  "integrity": "sha512-LLoxjALd+LRzpjhaH1N2m8n4lRabRL5OXKf7cbVw0/2eq0hG5+D8Uw7jFt+34a8vKjEUe4z/p348QW08CQ==",
  "license": "MIT",
  "dependencies": {
    "forwarded": "0.2.0",
    "ipaddr.js": "1.9.1"
  },
  "engines": {
    "node": ">= 0.10"
  }
},
"node_modules/pstree.remy": {
  "version": "1.1.8",
  "resolved": "https://registry.npmjs.org/pstree.remy/-/pstree.remy-1.1.8.tgz",
  "integrity": "sha512-olTJ68iVnOztWVpULbOwFaVcDJwW4Lp5X4AElnIkcL07Vb8kh9S5wq6HeMe/2wmt0G9LH5yYmIat4Nkqtw==",
  "dev": true,
  "license": "MIT"
},
"node_modules/punycode": {
  "version": "2.3.1",
  "resolved": "https://registry.npmjs.org/punycode/-/punycode-2.3.1.tgz",
  "integrity": "sha512-WYzuM4w4FrguxDkFzP1AdB2ufcTfmhfrjFHt94D8o3/uV6F/cJb6bPPpVtP3IZgNs78z3ZUPVYF1PRqM/uA==",
  "license": "MIT",
  "engines": {
    "node": ">=6"
  }
},
"node_modules/qs": {
  "version": "6.13.0",
  "resolved": "https://registry.npmjs.org/qs/-/qs-6.13.0.tgz",
  "integrity": "sha512-113Y708iH/5Dy617zZe5p+yVX8Wn7KHsPqo3w5U8J8CvT5FydkzXh5Y9P76D8KJ3v7H4aH0T1U88F02A==",
  "license": "BSD-3-Clause",
  "dependencies": {
    "side-channel": "^1.0.6"
  },
  "engines": {
    "node": ">=0.6"
  }
},
"funding": {
  "url": "https://github.com/sponsors/ljharb"
}
},
"node_modules/randombytes": {
  "version": "2.1.0",
  "resolved": "https://registry.npmjs.org/randombytes/-/randombytes-2.1.0.tgz",
  "integrity": "sha512-42V3sLOIj+L6yXrdKmfhXO1fj7zZHj9h2tZWpXsgY1dQrH1UYtPzqfESloLLH+zKYfy9XqYK5Sj1oQrP6Q==",
  "dev": true,
  "license": "MIT",
  "dependencies": {
    "safe-buffer": "^5.1.0"
  }
},
"node_modules/range-parser": {
  "version": "1.2.1",
  "resolved": "https://registry.npmjs.org/range-parser/-/range-parser-1.2.1.tgz",
  "integrity": "sha512-08y3urKxqZkf9HkEKLc9NoyQIhHSXx6Lwz/Sz8/BkyIC7a64Fq8BeWnXJdlFsLj/5iN8kEqj7Vrx9Cy8MjQ==",
  "license": "MIT",
  "engines": {
    "node": ">= 0.6"
  }
},
"node_modules/raw-body": {
  "version": "2.5.2",
  "resolved": "https://registry.npmjs.org/raw-body/-/raw-body-2.5.2.tgz",
  "integrity": "sha512-4zys6Xb1X43103G2u8Gc8f/2y5Bq1CafL4hE72jOsHHQKkXq/39MI8X94Pm1aA3SIFftHpwLrUyS9T8eQ==",
```

```

"license": "MIT",
"dependencies": {
  "bytes": "3.1.2",
  "http-errors": "2.0.0",
  "iconv-lite": "0.4.24",
  "unpipe": "1.0.0"
},
"engines": {
  "node": ">= 0.8"
}
},
"node_modules/readable-stream": {
  "version": "3.6.2",
  "resolved": "https://registry.npmjs.org/readable-stream/-/readable-stream-3.6.2.tgz",
  "integrity": "sha512-9u/sniCrY3D5WdsERHzHE4G2YCXqoG5FTHUICC4Sibr6XcLZBY05ya9EKjYek9O5xOAwjGq+1JdGBAS7Q9ScoA==",
  "license": "MIT",
  "dependencies": {
    "inherits": "^2.0.3",
    "string_decoder": "^1.1.1",
    "util-deprecate": "^1.0.1"
  },
  "engines": {
    "node": ">= 6"
  }
},
"node_modules/readdirp": {
  "version": "3.6.0",
  "resolved": "https://registry.npmjs.org/readdirp/-/readdirp-3.6.0.tgz",
  "integrity": "sha512-hOS089on8RduqdbhvQ5Z37A0ESjsqz6qnRcffsMU3495FuTdqSm+7bhJ29JvIOsBDEEan5DPu9t3To9VRIMzA==",
  "dev": true,
  "license": "MIT",
  "dependencies": {
    "picomatch": "^2.2.1"
  },
  "engines": {
    "node": ">=8.10.0"
  }
},
"node_modules/require-directory": {
  "version": "2.1.1",
  "resolved": "https://registry.npmjs.org/require-directory/-/require-directory-2.1.1.tgz",
  "integrity": "sha512-fGxEl7+wsG9xrvdjsrlmL22OMTTiHRwAMroiEeMgq8gzLC/PQr7RsRDSTLUg/bZAZtF+TVIkHc6/4RIKrui+Q==",
  "dev": true,
  "license": "MIT",
  "engines": {
    "node": ">=0.10.0"
  }
},
"node_modules/rimraf": {
  "version": "3.0.2",
  "resolved": "https://registry.npmjs.org/rimraf/-/rimraf-3.0.2.tgz",
  "integrity": "sha512-JZkIMZkAGFFPP2YqXZXPbMIMBgsxzE8ILs4lMIX/2o0L9UBw9O/Y3o6wFw/i9YLapcUJWwqbi3kdxlPdC62TIA==",
  "deprecated": "Rimraf versions prior to v4 are no longer supported",
  "license": "ISC",
  "dependencies": {
    "glob": "^7.1.3"
  },
  "bin": {
    "rimraf": "bin.js"
  },
  "funding": {
    "url": "https://github.com/sponsors/isaacs"
  }
},
"node_modules/safe-buffer": {
  "version": "5.2.1",
  "resolved": "https://registry.npmjs.org/safe-buffer/-/safe-buffer-5.2.1.tgz",
  "integrity": "sha512-RdEqBl5H9Xfrs9QvumfGCsS4U6uIMUohV8hP1EfTzeYRu9/5dY/zUeuzdsIKhhZFLZNM5BvYz1vjEHXzEA==",
  "funding": [
    {
      "type": "github",
      "url": "https://github.com/sponsors/feross"
    },
    {
      "type": "patreon",
      "url": "https://www.patreon.com/feross"
    }
  ],

```

```
{
  "type": "consulting",
  "url": "https://feross.org/support"
}
],
"license": "MIT"
},
"node_modules/safer-buffer": {
  "version": "2.1.2",
  "resolved": "https://registry.npmjs.org/safer-buffer/-/safer-buffer-2.1.2.tgz",
  "integrity": "sha512-Zo3K82SD7Riyi0E1EQPojLz7kpepnSQI9IyPbHHg1XXXevb5dJI7tpyN2ADxGcQbHG7vcyRHk0cbwqcQriUtg==",
  "license": "MIT"
},
"node_modules/semver": {
  "version": "7.6.3",
  "resolved": "https://registry.npmjs.org/semver/-/semver-7.6.3.tgz",
  "integrity": "sha512-oVekP1cKti+CTDvHWYFUcMtsK/00wmAEfyqKfNdARm8u1wNVhSgaX7A8d4UuIU5e84iEwOhs7ZPYRmzU9U6A==",
  "license": "ISC",
  "bin": {
    "semver": "bin/semver.js"
  },
  "engines": {
    "node": ">=10"
  }
},
"node_modules/send": {
  "version": "0.19.0",
  "resolved": "https://registry.npmjs.org/send/-/send-0.19.0.tgz",
  "integrity": "sha512-dW41u5VfLXu8SJh5bwRmyYUbaAoSB3c9uQh6L8h/KtsFREWPbX1lrIjJo186Jc4nmci/sGUZ9a0a0J2zgfq2hw==",
  "license": "MIT",
  "dependencies": {
    "debug": "2.6.9",
    "depd": "2.0.0",
    "destroy": "1.2.0",
    "encodeurl": "~1.0.2",
    "escape-html": "~1.0.3",
    "etag": "~1.8.1",
    "fresh": "0.5.2",
    "http-errors": "2.0.0",
    "mime": "1.6.0",
    "ms": "2.1.3",
    "on-finished": "2.4.1",
    "range-parser": "~1.2.1",
    "statuses": "2.0.1"
  },
  "engines": {
    "node": ">= 0.8.0"
  }
},
"node_modules/send/node_modules/encodeurl": {
  "version": "1.0.2",
  "resolved": "https://registry.npmjs.org/encodeurl/-/encodeurl-1.0.2.tgz",
  "integrity": "sha512-TPJXq8JqFaVYm2CWmPvnP2Iyo4ZSM7/QKcSmuMLDObfH5fi7RUGmd/rTDf+rut/saiDiQEeVTNgAmJEdAOx0w==",
  "license": "MIT",
  "engines": {
    "node": ">= 0.8"
  }
},
"node_modules/send/node_modules/ms": {
  "version": "2.1.3",
  "resolved": "https://registry.npmjs.org/ms/-/ms-2.1.3.tgz",
  "integrity": "sha512-6FlzubTLZG3J2a/NVCAleEhJzq55xgHyaCU9yYXvcLsvoVaHJq/s5xXI6/XXP6tz7R9xAOtHnSO/tXtF3WRTIA==",
  "license": "MIT"
},
"node_modules/serialize-javascript": {
  "version": "6.0.2",
  "resolved": "https://registry.npmjs.org/serialize-javascript/-/serialize-javascript-6.0.2.tgz",
  "integrity": "sha512-Saa1xPByTTq2gdeFZYLLo+RFE35NHZkAbqZeWNd3BpzppeVisAqpDjcp8dyf6ulvEqJRd46jemmyA4iFleVk8g==",
  "dev": true,
  "license": "BSD-3-Clause",
  "dependencies": {
    "randombytes": "^2.1.0"
  }
},
"node_modules/serve-static": {
  "version": "1.16.2",
```

```
"resolved": "https://registry.npmjs.org/serve-static/-/serve-static-1.16.2.tgz",
"integrity": "sha512-VqpjJZKadQB/PEbEwvFdO43Ax5dFBZ2UECszz8bQ7pi7wt/PWe1P6MN7eCnjsatYtBT6EuiClbjSWP2Wrl0Tw==",
"license": "MIT",
"dependencies": {
  "encodeurl": "~2.0.0",
  "escape-html": "~1.0.3",
  "parseurl": "~1.3.3",
  "send": "0.19.0"
},
"engines": {
  "node": ">= 0.8.0"
}
},
"node_modules/set-blocking": {
  "version": "2.0.0",
  "resolved": "https://registry.npmjs.org/set-blocking/-/set-blocking-2.0.0.tgz",
  "integrity": "sha512-KiKBS8AnWGEyLzofFfmvKwPdPzqiy16LvQfK3yv/fVH7Bj13/wl3JSR1J+rfgRE9q7xUJK4qvgS8raSOeLUehw==",
  "license": "ISC"
},
"node_modules/setprototypeof": {
  "version": "1.2.0",
  "resolved": "https://registry.npmjs.org/setprototypeof/-/setprototypeof-1.2.0.tgz",
  "integrity": "sha512-565637E8a211e7a04c73a89c42300937722e3868",
  "license": "ISC"
},
"node_modules/shebang-command": {
  "version": "2.0.0",
  "resolved": "https://registry.npmjs.org/shebang-command/-/shebang-command-2.0.0.tgz",
  "integrity": "sha512-kHxr2zZpYtdmrN1qDjrrX/Z1rR1kG8Dx+gkpK1G4eXmvXswmcE1hTWBWWYUzIraYw1/yZp6YuDY77YtvbN0dmDA==",
  "dev": true,
  "license": "MIT",
  "dependencies": {
    "shebang-regex": "^3.0.0"
  },
  "engines": {
    "node": ">=8"
  }
},
"node_modules/shebang-regex": {
  "version": "3.0.0",
  "resolved": "https://registry.npmjs.org/shebang-regex/-/shebang-regex-3.0.0.tgz",
  "integrity": "sha512-7wbn/qb1iA69FyYd4g5Jmtu3eP/VUFqO+YI13cUwO7jTPP841hT0bLz78yKqQ4FywhSod3XU5T432QkX3w==",
  "dev": true,
  "license": "MIT",
  "engines": {
    "node": ">=8"
  }
},
"node_modules/side-channel": {
  "version": "1.1.0",
  "resolved": "https://registry.npmjs.org/side-channel/-/side-channel-1.1.0.tgz",
  "integrity": "sha512-fV3jwX3E91D8G6ZCm3VJbDnK9s6kKQuwpoa1StQX7pND70+q50CLHZMpW1X4BKOAb+1J4OZP16HncE8A==",
  "license": "MIT",
  "dependencies": {
    "es-errors": "^1.3.0",
    "object-inspect": "^1.13.3",
    "side-channel-list": "^1.0.0",
    "side-channel-map": "^1.0.1",
    "side-channel-weakmap": "^1.0.2"
  },
  "engines": {
    "node": ">= 0.4"
  },
  "funding": {
    "url": "https://github.com/sponsors/ljharb"
  }
},
"node_modules/side-channel-list": {
  "version": "1.0.0",
  "resolved": "https://registry.npmjs.org/side-channel-list/-/side-channel-list-1.0.0.tgz",
  "integrity": "sha512-FV0eSxi2gcn/VBjImwgBwwvE4Ql135Tn17x9X8z7QVIM3lTWHJ/m32ozf38W5e8Hte11En511N+5dprzW8w==",
  "license": "MIT",
  "dependencies": {
    "es-errors": "^1.3.0",
    "object-inspect": "^1.13.3"
  }
}
```

```

    },
    "engines": {
      "node": ">= 0.4"
    },
    },
    "funding": {
      "url": "https://github.com/sponsors/ljharb"
    }
  },
  "node_modules/side-channel-map": {
    "version": "1.0.1",
    "resolved": "https://registry.npmjs.org/side-channel-map/-/side-channel-map-1.0.1.tgz",
    "integrity": "sha512-VcJCNfMsby3TDo02nbjTM/ewra6jPHmpThenkTYh8pG9ucZ/1P8So4u4FGBek/BjpOVsDCMoLA/iuBKIFXRA==",
    "license": "MIT",
    "dependencies": {
      "call-bound": "^1.0.2",
      "es-errors": "^1.3.0",
      "get-intrinsic": "^1.2.5",
      "object-inspect": "^1.13.3"
    },
    "engines": {
      "node": ">= 0.4"
    },
    },
    "funding": {
      "url": "https://github.com/sponsors/ljharb"
    }
  },
  "node_modules/side-channel-weakmap": {
    "version": "1.0.2",
    "resolved": "https://registry.npmjs.org/side-channel-weakmap/-/side-channel-weakmap-1.0.2.tgz",
    "integrity": "sha512-WPS/HvHQTYnHisLo9McqBHOJk2FkHO/tlpvldyrnem4aeQp4hai3gythswg6p01oSoTL58rcpiFAjF2br2Ak2A==",
    "license": "MIT",
    "dependencies": {
      "call-bound": "^1.0.2",
      "es-errors": "^1.3.0",
      "get-intrinsic": "^1.2.5",
      "object-inspect": "^1.13.3",
      "side-channel-map": "^1.0.1"
    },
    "engines": {
      "node": ">= 0.4"
    },
    },
    "funding": {
      "url": "https://github.com/sponsors/ljharb"
    }
  },
  "node_modules/sift": {
    "version": "17.1.3",
    "resolved": "https://registry.npmjs.org/sift/-/sift-17.1.3.tgz",
    "integrity": "sha512-Rltj66/bOlCeFzYTUNvX/EF1igRbbnGSvEyT79McoZa/DeGhMyC5pWKOEsZKnkqtSeovd5FL/bjHWC3CllvCQ==",
    "license": "MIT"
  },
  "node_modules/signal-exit": {
    "version": "3.0.7",
    "resolved": "https://registry.npmjs.org/signal-exit/-/signal-exit-3.0.7.tgz",
    "integrity": "sha512-dk2lu7a4j8f149e1M0WJpRNtH0vda53Bb8d8Hk4EzPWNrH61H817KQEuHxbGQ1Qr26h8vT5wP85uNIoQ=="
  },
  "node_modules/sponspc/uoS0vXeE9L1+0wuaMqKlfz9AMUo38JsyLSBWSFcHR1Rri62LZc12vLr1gb3jl7iwQhgwAbGQ==",
    "license": "ISC"
  },
  "node_modules/simple-update-notifier": {
    "version": "2.0.0",
    "resolved": "https://registry.npmjs.org/simple-update-notifier/-/simple-update-notifier-2.0.0.tgz",
    "integrity": "sha512-2w9sD9IXB5/4y17OnGs6D084liYMBQ5uWApN1qFWnNCt1I1K63nlP29u5fD9QZGvzV1B5SlTzJvU1NpUjg==",
    "dev": true,
    "license": "MIT",
    "dependencies": {
      "semver": "^7.5.3"
    },
    "engines": {
      "node": ">=10"
    }
  },
  "node_modules/sinon": {
    "version": "19.0.2",
    "resolved": "https://registry.npmjs.org/sinon/-/sinon-19.0.2.tgz",
    "integrity": "sha512-euuToqM+PjO4UgXeLEtsfQiuoyPXlqFezr6YZDFwHR3t4qaX0fZUe1MfPMznTL5f8BWRVS89KduLdMUxFCO6g==",
    "dev": true,
    "license": "BSD-3-Clause",

```

```
"dependencies": {
  "@sinonjs/commons": "^3.0.1",
  "@sinonjs/fake-timers": "^13.0.2",
  "@sinonjs/samsam": "^8.0.1",
  "diff": "^7.0.0",
  "nise": "^6.1.1",
  "supports-color": "^7.2.0"
},
"funding": {
  "type": "opencollective",
  "url": "https://opencollective.com/sinon"
}
},
"node_modules/sinon/node_modules/diff": {
  "version": "7.0.0",
  "resolved": "https://registry.npmjs.org/diff/-/diff-7.0.0.tgz",
  "integrity": "sha512-PJWHUUb1RFevKCwaFA9RLG5tCd+FO5iRh9A8HEtkmBH2Li03iJriB6m6JIN4rGz3K3JLawl7/veA1xzRKP6ISBw==",
  "dev": true,
  "license": "BSD-3-Clause",
  "engines": {
    "node": ">=0.3.1"
  }
},
"node_modules/sinon/node_modules/has-flag": {
  "version": "4.0.0",
  "resolved": "https://registry.npmjs.org/has-flag/-/has-flag-4.0.0.tgz",
  "integrity": "sha512-EykJT/Q1KJTWctppgIAgfSO0tkVuZUjhGMr17kqTumMl6Afv3EISleU7qZUzoXDFTAHTDC4NOoG/ZxU3EvlMPQ==",
  "dev": true,
  "license": "MIT",
  "engines": {
    "node": ">=8"
  }
},
"node_modules/sinon/node_modules/supports-color": {
  "version": "7.2.0",
  "resolved": "https://registry.npmjs.org/supports-color/-/supports-color-7.2.0.tgz",
  "integrity": "sha512-9o91t2850352332e9784o624b3L6871bW0+5L56o228P78c78499u9qf8qY6N9tr1jP912ct1wqlfO+Lwxc9+0q2wLyy4Dfvt80/DPn2pqOBsJdDiogXGR9+OvwRw==",
  "dev": true,
  "license": "MIT",
  "dependencies": {
    "has-flag": "^4.0.0"
  },
  "engines": {
    "node": ">=8"
  }
},
"node_modules/sparse-bitfield": {
  "version": "3.0.3",
  "resolved": "https://registry.npmjs.org/sparse-bitfield/-/sparse-bitfield-3.0.3.tgz",
  "integrity": "sha512-kvzhi7vqKtFkh0PZU+2D2Pillw2ymqJKujUcyPMd9Y75Nv4nPbGJZXNhxsgdQab2BmlDct1YnfQCguEvHr7VsQ==",
  "license": "MIT",
  "dependencies": {
    "memory-pager": "^1.0.2"
  }
},
"node_modules/statuses": {
  "version": "2.0.1",
  "resolved": "https://registry.npmjs.org/statuses/-/statuses-2.0.1.tgz",
  "integrity": "sha512-9339/7Y3KzQa30wJ3YkD4j2wQ+lFtTJ4v4b98n0wvowdVlRgUGIeD/VR6z3S5npwOp7+1wQhIKDkz5Q==",
  "license": "MIT",
  "engines": {
    "node": ">= 0.8"
  }
},
"node_modules/string_decoder": {
  "version": "1.3.0",
  "resolved": "https://registry.npmjs.org/string_decoder/-/string_decoder-1.3.0.tgz",
  "integrity": "sha512-hjqPYjp1oXilNW3Y6h90hCgD1tEfXgVq8fZysqBNMD59v8a8/BdYKdOaGp1Gj8Q5wYcWUHdBJA8GtS+o==",
  "license": "MIT",
  "dependencies": {
    "safe-buffer": "~5.2.0"
  }
},
"node_modules/string-width": {
  "version": "4.2.3",
```

```
"resolved": "https://registry.npmjs.org/string-width/-/string-width-4.2.3.tgz",
"integrity": "sha512-wKyQRQpjJ0sLP62ErSZdGsjMJWsap5oRNihHhu6G7JVO/9jIB6UyevL+tXuOqrng8j/cxKTWyuWUwvSTriiZz/g==",
"license": "MIT",
"dependencies": {
  "emoji-regex": "^8.0.0",
  "is-fullwidth-code-point": "^3.0.0",
  "strip-ansi": "^6.0.1"
},
"engines": {
  "node": ">=8"
}
},
"node_modules/string-width-cjs": {
  "name": "string-width",
  "version": "4.2.3",
  "resolved": "https://registry.npmjs.org/string-width/-/string-width-4.2.3.tgz",
  "integrity": "sha512-wKyQRQpjJ0sLP62ErSZdGsjMJWsap5oRNihHhu6G7JVO/9jIB6UyevL+tXuOqrng8j/cxKTWyuWUwvSTriiZz/g==",
  "dev": true,
  "license": "MIT",
  "dependencies": {
    "emoji-regex": "^8.0.0",
    "is-fullwidth-code-point": "^3.0.0",
    "strip-ansi": "^6.0.1"
  },
  "engines": {
    "node": ">=8"
  }
},
"node_modules/strip-ansi": {
  "version": "6.0.1",
  "resolved": "https://registry.npmjs.org/strip-ansi/-/strip-ansi-6.0.1.tgz",
  "integrity": "sha512-Y38VPSHcQkFrCpFnQ9vuSXmquuv5oXOKpGeT6aGrr3o3Gc9AlVa6JBfUSOCnbxGGZF+/0oolI7KrPuUSztUdU5A==",
  "license": "MIT",
  "dependencies": {
    "ansi-regex": "^5.0.1"
  },
  "engines": {
    "node": ">=8"
  }
},
"node_modules/strip-ansi-cjs": {
  "name": "strip-ansi",
  "version": "6.0.1",
  "resolved": "https://registry.npmjs.org/strip-ansi/-/strip-ansi-6.0.1.tgz",
  "integrity": "sha512-Y38VPSHcQkFrCpFnQ9vuSXmquuv5oXOKpGeT6aGrr3o3Gc9AlVa6JBfUSOCnbxGGZF+/0oolI7KrPuUSztUdU5A==",
  "dev": true,
  "license": "MIT",
  "dependencies": {
    "ansi-regex": "^5.0.1"
  },
  "engines": {
    "node": ">=8"
  }
},
"node_modules/strip-json-comments": {
  "version": "3.1.1",
  "resolved": "https://registry.npmjs.org/strip-json-comments/-/strip-json-comments-3.1.1.tgz",
  "integrity": "sha512-6fPc+R4ihwqP6N/alv2f1gMH8lOVtWQHocC4yK6oSDVvOcunAsfCqjkXnqiYMhmMwS/mEHLp7Vehlt3ql6lEig==",
  "dev": true,
  "license": "MIT",
  "engines": {
    "node": ">=8"
  },
  "funding": {
    "url": "https://github.com/sponsors/sindresorhus"
  }
},
"node_modules/supports-color": {
  "version": "5.5.0",
  "resolved": "https://registry.npmjs.org/supports-color/-/supports-color-5.5.0.tgz",
  "integrity": "sha512-QjVjwdXlt408MliAqCX4oUKsgU2EqAGzs2Ppkkm4AQYbjm+ZEWECW4SfFNTR4uMNZma0ey4f5lglRkBoaX0QMow==",
  "dev": true,
  "license": "MIT",
  "dependencies": {
    "has-flag": "^3.0.0"
  },
  "engines": {
```

```
    "node": ">=4"
  },
  "node_modules/tar": {
    "version": "6.2.1",
    "resolved": "https://registry.npmjs.org/tar/-/tar-6.2.1.tgz",
    "integrity": "sha512-DZ4yORTwrbTj/7MZyq2w+/ZFdl6OZ/f9SFHR+71glVUZhOQPHzVCLPvRnPgYaMpfWxxk/4ONva3GQSyNlKRv6A==",
    "license": "ISC",
    "dependencies": {
      "chownr": "^2.0.0",
      "fs-minipass": "^2.0.0",
      "minipass": "^5.0.0",
      "minizlib": "^2.1.1",
      "mkdirp": "^1.0.3",
      "yallist": "^4.0.0"
    },
    "engines": {
      "node": ">=10"
    }
  },
  "node_modules/to-regex-range": {
    "version": "5.0.1",
    "resolved": "https://registry.npmjs.org/to-regex-range/-/to-regex-range-5.0.1.tgz",
    "integrity": "sha512-65P7iz6X5yEr1cwgcgvQxbblw7Uk3gOy5dIdtZ4rDveLqhrdJP+Li/Hx6tyK0NEb+2GCyneCMJiGqrADCSNk8sQ==",
    "dev": true,
    "license": "MIT",
    "dependencies": {
      "is-number": "^7.0.0"
    },
    "engines": {
      "node": ">=8.0"
    }
  },
  "node_modules/toidentifier": {
    "version": "1.0.1",
    "resolved": "https://registry.npmjs.org/toidentifier/-/toidentifier-1.0.1.tgz",
    "integrity": "sha512-o5sSPKEkg/DIQNmH43V0/uerLrpzVedkUh8tGNvaeXpfpuwjKenlSox/2O/BTIZUtEe+JG7s5YhEz608PLAHRA==",
    "license": "MIT",
    "engines": {
      "node": ">=0.6"
    }
  },
  "node_modules/touch": {
    "version": "3.1.1",
    "resolved": "https://registry.npmjs.org/touch/-/touch-3.1.1.tgz",
    "integrity": "sha512-4w0oP288iR33QY23WqTqUq3wvU013WJ1U+VY09V1+n/k2/86r5YU8VH4H+RXZ9X6V10ZtKq6XWqo8aA==",
    "dev": true,
    "license": "ISC",
    "bin": {
      "nodetouch": "bin/nodetouch.js"
    }
  },
  "node_modules/tr46": {
    "version": "5.0.0",
    "resolved": "https://registry.npmjs.org/tr46/-/tr46-5.0.0.tgz",
    "integrity": "sha512-8GfU7rvdVbkD62d3a435XnXUzY8l3oq8p5hFkq4LQLh9wEqCgjbqXZd96Z/8t1n3S3gk/wW0P3g+cA==",
    "license": "MIT",
    "dependencies": {
      "punycode": "^2.3.1"
    },
    "engines": {
      "node": ">=18"
    }
  },
  "node_modules/type-detect": {
    "version": "4.0.8",
    "resolved": "https://registry.npmjs.org/type-detect/-/type-detect-4.0.8.tgz",
    "integrity": "sha512-0v7+0o4/z8ySmrhX55H7w382sJenK0JLDeNoBJ9L++qkXcdTu3WgYg9HzLH+cdWub1EroL38b89Se8A==",
    "dev": true,
    "license": "MIT",
    "engines": {
      "node": ">=4"
    }
  },
  "node_modules/type-is": {
```

```

"version": "1.6.18",
"resolved": "https://registry.npmjs.org/type-is/-/type-is-1.6.18.tgz",
"integrity": "sha512-TkRKr9sUTxEH8MdfuGSP7VizJyzRNMjj2J2do2Jr3Kym598JVdEksuzPQCnIFPW4ky9Q+iA+ma9BGm06XQBy8g==",
"license": "MIT",
"dependencies": {
  "media-typer": "0.3.0",
  "mime-types": "~2.1.24"
},
"engines": {
  "node": ">= 0.6"
}
},
"node_modules/undefsafe": {
"version": "2.0.5",
"resolved": "https://registry.npmjs.org/undefsafe/-/undefsafe-2.0.5.tgz",
"integrity": "sha512-WxONCrssBM8TSPRqN5EmsjVrsv4A8X12J4ArBiiayv3DyyG3Zllg6yysuuSYdZsVz3TKcTg2fd//Ujd4CHV1iA==",
"dev": true,
"license": "MIT"
},
"node_modules/unpipe": {
"version": "1.0.0",
"resolved": "https://registry.npmjs.org/unpipe/-/unpipe-1.0.0.tgz",
"integrity": "sha512-pjy2bYhSsufwWlKwPc+l3cN7+wwJK6uz0YdJEOLQDbl6jo/YlPi4mb8agUkVC8BF7V8NuzeyPNqRksA3hztKQ==",
"license": "MIT",
"engines": {
  "node": ">= 0.8"
}
},
"node_modules/util-deprecate": {
"version": "1.0.2",
"resolved": "https://registry.npmjs.org/util-deprecate/-/util-deprecate-1.0.2.tgz",
"integrity": "sha512-EPD5q1uXyFxJpCrLnCc1nHnq3gOa6DZBocAlil2TaSCA7VCJ1UJDMagCzIkXNsUYfD1daK//LTEQ8xilbrHtcw==",
"license": "MIT"
},
"node_modules/utlils-merge": {
"version": "1.0.1",
"resolved": "https://registry.npmjs.org/utlils-merge/-/utlils-merge-1.0.1.tgz",
"integrity": "sha512-pMZTvlkT1d+TFGvDOqodOclx0QWkkgi6Tdoa8gC8ffGAAqz9pzPTZWAYbbsHHoED/zTMtkv/VoYTYyShUn81hA==",
"license": "MIT",
"engines": {
  "node": ">= 0.4.0"
}
},
"node_modules/vary": {
"version": "1.1.2",
"resolved": "https://registry.npmjs.org/vary/-/vary-1.1.2.tgz",
"integrity": "sha512-BNGbWlfd0eUPabhkXUVm0j8uuvREyTh5ovRa/dyow/BqAbZJyC+5fU+IzQOzmAKzYqYRAISoRhdQr3eIZ/PXqg==",
"license": "MIT",
"engines": {
  "node": ">= 0.8"
}
},
"node_modules/webidl-conversions": {
"version": "7.0.0",
"resolved": "https://registry.npmjs.org/webidl-conversions/-/webidl-conversions-7.0.0.tgz",
"integrity": "sha512-VWdDukDzu71offAQR975unBIGqfKZpM+8ZX6ySk8nYhVoo5CYaZyzt3YBvYtRtO+aoGlqxPg/B87NGVZ/fu6g==",
"license": "BSD-2-Clause",
"engines": {
  "node": ">=12"
}
},
"node_modules/whatwg-url": {
"version": "14.1.0",
"resolved": "https://registry.npmjs.org/whatwg-url/-/whatwg-url-14.1.0.tgz",
"integrity": "sha512-jltf/foYlKywAt3x/XWKZ/3rz8OSJPIWktjmk891aJUUEjiVxKX9LEO92qH3hv4aJ0mN3MWPvGMCy8jQi95xK4w==",
"license": "MIT",
"dependencies": {
  "tr46": "^5.0.0",
  "webidl-conversions": "^7.0.0"
},
"engines": {
  "node": ">=18"
}
}

```

```
  },
  "node_modules/which": {
    "version": "2.0.2",
    "resolved": "https://registry.npmjs.org/which/-/which-2.0.2.tgz",
    "integrity": "sha512-BLI3Tl1TW3Pvl70l3yq3Y64i+awpwxqsGBYWkkqMtnbXgrMD+yj7rhW0kuEDxJaYXGjEW5ogapKNMEKNMjibA==",
    "dev": true,
    "license": "ISC",
    "dependencies": {
      "isexe": "^2.0.0"
    }
  },
  "bin": {
    "node-which": "bin/node-which"
  },
  "engines": {
    "node": ">= 8"
  }
},
"node_modules/wide-align": {
  "version": "1.1.5",
  "resolved": "https://registry.npmjs.org/wide-align/-/wide-align-1.1.5.tgz",
  "integrity": "sha512-J+zNcLH8C5e28JF1aQOxnuAiqsCfFH9vSRyFKoJ1zXjNfywG1VX44jqGwOp60YUbe1C3XTZRdyGyDh9S0t9Ng==",
  "license": "ISC",
  "dependencies": {
    "string-width": "^1.0.2 || 2 || 3 || 4"
  }
},
"node_modules/workerpool": {
  "version": "6.5.1",
  "resolved": "https://registry.npmjs.org/workerpool/-/workerpool-6.5.1.tgz",
  "integrity": "sha512-FU4U9aSS/8wJUM0VCvUY9E0WTOXi9p8k1vzA8ySe6X40UqXQ3Vp8+7x11pCZnKyA6V8mU03PlyuV1qX5Fg==",
  "dev": true,
  "license": "Apache-2.0"
},
"node_modules/wrap-ansi": {
  "version": "7.0.0",
  "resolved": "https://registry.npmjs.org/wrap-ansi/-/wrap-ansi-7.0.0.tgz",
  "integrity": "sha512-YVGIj2kamLSTxw6NsZjoBxfSwsn0ycdesmc4p+Q21c5zPuZ1pl+NfxVdxPtdHvmNVOQ6XSYG4AUtyt/Fi7D16Q==",
  "dev": true,
  "license": "MIT",
  "dependencies": {
    "ansi-styles": "^4.0.0",
    "string-width": "^4.1.0",
    "strip-ansi": "^6.0.0"
  },
  "engines": {
    "node": ">=10"
  },
  "funding": {
    "url": "https://github.com/chalk/wrap-ansi?sponsor=1"
  }
},
"node_modules/wrap-ansi-cjs": {
  "name": "wrap-ansi",
  "version": "7.0.0",
  "resolved": "https://registry.npmjs.org/wrap-ansi/-/wrap-ansi-7.0.0.tgz",
  "integrity": "sha512-YVGIj2kamLSTxw6NsZjoBxfSwsn0ycdesmc4p+Q21c5zPuZ1pl+NfxVdxPtdHvmNVOQ6XSYG4AUtyt/Fi7D16Q==",
  "dev": true,
  "license": "MIT",
  "dependencies": {
    "ansi-styles": "^4.0.0",
    "string-width": "^4.1.0",
    "strip-ansi": "^6.0.0"
  },
  "engines": {
    "node": ">=10"
  },
  "funding": {
    "url": "https://github.com/chalk/wrap-ansi?sponsor=1"
  }
},
"node_modules/wrappy": {
  "version": "1.0.2",
  "resolved": "https://registry.npmjs.org/wrappy/-/wrappy-1.0.2.tgz",
  "integrity": "sha512-l4Sp/DRseor9wL6EvV2+TuQn63dMkPjZ/sp9XkghtTEbV99KIPSIxUsZ3u7/IQO4wxtcFB4bgpQPRcR3QCvezPcQ==",
  "license": "ISC"
},
}
```

```

"node_modules/y18n": {
  "version": "5.0.8",
  "resolved": "https://registry.npmjs.org/y18n/-/y18n-5.0.8.tgz",
  "integrity": "sha512-0pPfZgeDWJHJIAmTLRP2DwHjdF5s7jo9tuztdQxAhINCdvS+3nGINqPd00AphqJR/0LhANUS6/+7SCb98YOFA==",
  "dev": true,
  "license": "ISC",
  "engines": {
    "node": ">=10"
  }
},
"node_modules/yallist": {
  "version": "4.0.0",
  "resolved": "https://registry.npmjs.org/yallist/-/yallist-4.0.0.tgz",
  "integrity": "sha512-3wdGidZyq5PB084XLES5TpOSRA3wjXAlIWMhum2kRcv/41Sn2emQ0dycQW4uZXLejwKvg6EsvbdlVL+FYEct7A==",
  "license": "ISC"
},
"node_modules/yargs": {
  "version": "17.7.2",
  "resolved": "https://registry.npmjs.org/yargs/-/yargs-17.7.2.tgz",
  "integrity": "sha512-0dK309w38C3ZB0pK1gH5C4Zb03tq8kQ1u0Y164i1Nvtrnbo0Y17dTc1rO1vEhUFBzKBzD85Vqo2H78A==",
  "dev": true,
  "license": "MIT",
  "dependencies": {
    "cliui": "^8.0.1",
    "escalade": "^3.1.1",
    "get-caller-file": "^2.0.5",
    "require-directory": "^2.1.1",
    "string-width": "^4.2.3",
    "y18n": "^5.0.5",
    "yargs-parser": "^21.1.1"
  },
  "engines": {
    "node": ">=12"
  }
},
"node_modules/yargs-parser": {
  "version": "21.1.1",
  "resolved": "https://registry.npmjs.org/yargs-parser/-/yargs-parser-21.1.1.tgz",
  "integrity": "sha512-tVpsJW7DdjecAifPbblB1e3qxIQsE6NoPc5/eTdrbbIC4h0LVsWhnoa3g+m2HclBlujHzsxZ4VJVA+GUuc2/LBw==",
  "dev": true,
  "license": "ISC",
  "engines": {
    "node": ">=12"
  }
},
"node_modules/yargs-unparser": {
  "version": "2.0.0",
  "resolved": "https://registry.npmjs.org/yargs-unparser/-/yargs-unparser-2.0.0.tgz",
  "integrity": "sha512-13Qx4X6X4xjxkz1285MO0Xg6i84L9hYmZMZ0d7i4Ta1j73o6LEwLxhYZsyuazJgImlqs4omNs1Vo624A==",
  "dev": true,
  "license": "MIT",
  "dependencies": {
    "camelcase": "^6.0.0",
    "decamelize": "^4.0.0",
    "flat": "^5.0.2",
    "is-plain-obj": "^2.1.0"
  },
  "engines": {
    "node": ">=10"
  }
},
"node_modules/yocto-queue": {
  "version": "0.1.0",
  "resolved": "https://registry.npmjs.org/yocto-queue/-/yocto-queue-0.1.0.tgz",
  "integrity": "sha512-pVds1xhJG9xN4QD89bG9VRj89HqW8YBq7tOwNy5arL3kR4P6nDNDJ3rG6QpYsZM6uzm/1rQ8VUwJ98A==",
  "dev": true,
  "license": "MIT",
  "engines": {
    "node": ">=10"
  },
  "funding": {
    "url": "https://github.com/sponsors/sindresorhus"
  }
},
"services/auth-service": {

```

```

"version": "1.0.0",
"license": "ISC",
"dependencies": {
  "bcrypt": "^5.1.1",
  "bcryptjs": "^2.4.3",
  "body-parser": "^1.20.3",
  "dotenv": "^16.4.7",
  "express": "^4.21.2",
  "jsonwebtoken": "^9.0.2",
  "mongoose": "^8.9.5"
}
},
"services/patient-treatment": {
  "version": "1.0.0",
  "extraneous": true,
  "license": "ISC",
  "dependencies": {
    "axios": "^1.7.9",
    "cors": "^2.8.5",
    "dotenv": "^16.4.7",
    "express": "^4.21.2",
    "jsonwebtoken": "^9.0.2",
    "mongoose": "^8.9.6"
  }
}
}
}
}

```

---

g. TCH – PIS – Main / sonar-project.properties

```

# Unique key for your project
sonar.projectKey=TCH-PIS

# Display name for your project in SonarQube
sonar.projectName=TCH-PIS

# Version of the project
sonar.projectVersion=1.0

# Define the location of source files (pointing to your microservices code)
sonar.sources=services

# Specify the language (JavaScript for a Node.js project)
sonar.language=js

# URL of the SonarQube server (adjust if running on a different host/port)
sonar.host.url=http://localhost:9000

# Authentication token generated in SonarQube (replace YOUR_SONAR_TOKEN with the actual token)
sonar.login=Ysqa_b4f400d15864bc8e0399232ae72fbb4751b066d3

```

---

## 2. Shared

### a. TCH – PIS – Main / shared / db.js

```
const mongoose = require('mongoose');
require('dotenv').config();

const connectDB = async () => {
  try {
    // await mongoose.connect(process.env.MONGO_URI, {
    //   useNewUrlParser: true,
    //   useUnifiedTopology: true
    // });
    await mongoose.connect(process.env.MONGO_URI);
    console.log('MongoDB connected');
  } catch (err) {
    console.error('Database connection failed:', err.message);
    process.exit(1); // Exit process if connection fails
  }
};

// Check if the connection was successful
mongoose.connection.on('connected', () => {
  console.log('Mongoose is connected to the database');
});

mongoose.connection.on('error', (err) => {
  console.error('Error with MongoDB connection:', err.message);
});

module.exports = connectDB;
```

### b. TCH – PIS – Main / shared / counterModel.js

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

const counterSchema = new mongoose.Schema({
  model: { type: String, required: true, unique: true }, // eg: 'patient', 'doctor'
  count: { type: Number, default: 0 }
});

module.exports = mongoose.model('Counter', counterSchema);
```

### 3. services

#### a. auth-service

##### i. TCH – PIS – Main / services / auth-service / src / controllers / authController.js

```
const User = require('../models/User');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');

//User Registration
exports.register= async (req, res) =>{
  try{
    const { username, password, role } = req.body;

    // Check for missing required fields
    if (!username || !password || !role) {
      return res.status(400).json({ message: 'Missing required fields' });
    }

    // Check if user already exists
    const existingUser = await User.findOne({ username });

    if (existingUser) {
      return res.status(400).json({ message: 'Username already exists' });
    }

    // Hash the password
    const hashedPassword = await bcrypt.hash(password, 10);

    // Create new user
    const newUser = new User({
      username,
      password: hashedPassword,
      role
    });

    // Save user to database
    await newUser.save();
    res.status(201).json({ message: 'User registered successfully' });
  }
  catch (error) {
    res.status(500).json({ message: 'Error registering user', error: error.message });
  }
};

//User Login
exports.login = async (req, res) =>{
  try{
    const { username, password } = req.body;

    // Find user by username
    const user = await User.findOne({ username });
    if (!user) {
      return res.status(401).json({ message: 'Invalid credentials' });
    }

    // Check password
    const isPasswordValid = await bcrypt.compare(password, user.password);
    if (!isPasswordValid) {
      return res.status(401).json({ message: 'Invalid credentials' });
    }

    // Generate JWT token
    const token = jwt.sign(
      { userId: user._id, role: user.role },
      process.env.JWT_SECRET,
      { expiresIn: '1h' }
    );

    // Set status to 200 and send the response
    return res.status(200).json({ token, role: user.role });
  }
  catch (error) {
    res.status(500).json({ message: 'Error logging in', error: error.message });
  }
};
```

ii. TCH – PIS – Main / services / auth-service / src / models / User.js

```
//Generating schema
const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({
  username: {
    type: String,
    required: true,
    unique: true
  },
  password: {
    type: String,
    required: true
  },
  role: {
    type: String,
    enum: ['clerk', 'doctor', 'nurse', 'paramedic'],
    required: true
  },
  createdAt: {
    type: Date,
    default: Date.now
  }
});

module.exports = mongoose.model('User', userSchema);
```

iii. TCH – PIS – Main / services / auth-service / src / routes / authRoutes.js

```
const express = require('express');
const router = express.Router();
const authController = require('../controllers/authController');

router.post('/register', authController.register);
router.post('/login', authController.login);

module.exports = router;
```

iv. TCH – PIS – Main / services / auth-service / src / server.js

```
const express = require('express');
const mongoose = require('mongoose');
const connectDB = require('../../shared/db');
const authRoutes = require('../routes/authRoutes');
// require('dotenv').config({ path: '../.env' });
const dotenv = require('dotenv');
const path = require('path');
const envPath = path.resolve(__dirname, '../.env'); // Ensure correct path
console.log("Loading .env from:", envPath); // Debugging output
dotenv.config({ path: envPath });
const app = express();
// Middleware
app.use(express.json()); // This allows us to parse JSON bodies in requests

console.log("MONGO_URI from .env:", process.env.MONGO_URI); // Debugging line
// Connect to MongoDB
connectDB();
//Routes
app.use('/auth', authRoutes);
// Add a new route to insert a record
app.post('/add-record', async (req, res) => {
  try {
    const collection = mongoose.connection.db.collection('test_collection');
    const result = await collection.insertOne(req.body);
    res.status(201).json({ message: 'Record inserted', insertedId: result.insertedId });
  } catch (err) {
    console.error(err);
    res.status(500).json({ message: 'Failed to insert record', error: err.message });
  }
});
// Start the server
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

```

// 1. Import required libraries
const chai = require('chai');
const sinon = require('sinon');
const User = require('../src/models/User');
const authController = require('../src/controllers/authController');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');

// 2. Configure Chai for assertions
const expect = chai.expect;

// 3. Describe the test suite for the Auth Controller's register function
describe('Auth Controller - Register - Unit - Test', () => {
  // 4. After each test, restore all stubs to their original state
  afterEach(() => {
    sinon.restore();
  });

  // ----- Test Case: 1 -----
  // 5. Define a test case for successful user registration
  it('should register a new user', async () => {
    // 6. Mock the request object (req)
    const req = {
      body: {
        username: 'testuser',
        password: 'testpassword',
        role: 'clerk',
      },
    };

    // 7. Mock the response object (res)
    const res = {
      status: sinon.stub().returnsThis(), // Stub res.status to return the response object
      json: sinon.stub(), // Stub res.json to capture the response
    };

    // 8. Stub User.findOne to simulate no existing user
    sinon.stub(User, 'findOne').resolves(null);

    // 9. Stub bcrypt.hash to simulate password hashing
    sinon.stub(bcrypt, 'hash').resolves('hashedpassword');

    // 10. Stub User.prototype.save to simulate saving a user to the database
    sinon.stub(User.prototype, 'save').resolves();

    // 11. Call the register function from authController
    await authController.register(req, res);

    // 12. Assertions to verify the function's behavior
    expect(res.status.calledWith(201)).to.be.true; // Check if status 201 was sent
    expect(res.json.calledWith({ message: 'User registered successfully' })).to.be.true; // Check if the correct message was sent
  });

  // ----- Test Case: 2 -----
  it('should return an error if username already exists', async () => {
    {
      const req = {
        body: {
          username: 'existinguser',
          password: 'testpassword',
          role: 'clerk',
        },
      };
      const res = {
        status: sinon.stub().returnsThis(),
        json: sinon.stub(),
      };
      sinon.stub(User, 'findOne').resolves({ username: 'existinguser' });
      await authController.register(req, res);
      expect(res.status.calledWith(400)).to.be.true;
      expect(res.json.calledWith({ message: 'Username already exists' })).to.be.true;
    }
  });
});

```

```
// ----- Test Case: 3 -----

it('should return an error if required fields are missing', async () => {
  const req={
    body:{
      username:"",
      password:"",
      role:"",

    },
  };
  const res={
    status:sinon.stub().returnsThis(),
    json: sinon.stub(),
  };

  await authController.register(req, res);

  expect(res.status.calledWith(400)).to.be.true;
  expect(res.json.calledWith({ message: 'Missing required fields' })).to.be.true;

});

// ----- Test Case: 4 -----

it('should return an error if role is invalid', async () => {
  const req = {
    body: {
      username: 'testuser',
      password: 'testpassword',
      role: 'testrole', // Invalid role
    },
  };
  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub(),
  };

  // Stub User.findOne to simulate no existing user
  sinon.stub(User, 'findOne').resolves(null);

  // Stub bcrypt.hash to simulate password hashing
  sinon.stub(bcrypt, 'hash').resolves('hashedpassword');

  // Create a validation error that MongoDB would throw for invalid role
  const validationError = new Error("User validation failed: role: `testrole` is not a valid enum value for path `role`.");

  // Stub User.prototype.save to reject with the validation error
  sinon.stub(User.prototype, 'save').rejects(validationError);

  await authController.register(req, res);

  expect(res.status.calledWith(500)).to.be.true;
  expect(res.json.calledWith(sinon.match({
    message: 'Error registering user',
    error: sinon.match.string
  }))).to.be.true;
});

// ----- Test Case - 5 -----

it('should return an error if username already exists', async () => {
  // Stub the User.findOne method
  const findOneStub = sinon.stub(User, 'findOne').resolves({ username: 'existinguser' });

  const req = {
    body: {
      username: 'existinguser',
      password: 'testpassword',
      role: 'clerk',
    },
  };
  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub(),
  };

  await authController.register(req, res);

  expect(res.status.calledWith(400)).to.be.true;
  expect(res.json.calledWith({ message: 'Username already exists' })).to.be.true;
}
```

```

    // Restore the stubbed method
    findOneStub.restore();
  });

});

describe('Auth Controller - Login - Unit - Test', () => {
  afterEach(() => {
    sinon.restore();
  });

  // ----- Test Case: 1 -----
  it('should log in a user with valid credentials', async () => {
    const req = {
      body: {
        username: 'testuser',
        password: 'testpassword',
      },
    };
    const res = {
      status: sinon.stub().returnsThis(),
      json: sinon.stub(),
    };

    // Stub User.findOne to simulate finding the user
    const findOneStub = sinon.stub(User, 'findOne').resolves({
      _id: 'userId',
      username: 'testuser',
      password: 'hashedpassword',
      role: 'clerk',
    });

    // Stub bcrypt.compare to simulate password comparison
    const compareStub = sinon.stub(bcrypt, 'compare').resolves(true);

    // Stub jwt.sign to simulate token generation
    const signStub = sinon.stub(jwt, 'sign').returns('fakeToken');

    await authController.login(req, res);

    expect(res.status.calledWith(200)).to.be.true;
    expect(res.json.calledWith({ token: 'fakeToken', role: 'clerk' }))).to.be.true;

    // Restore the stubbed methods
    findOneStub.restore();
    compareStub.restore();
    signStub.restore();
  });

  // ----- Test Case: 2 -----
  it('should return an error if username is not found', async () => {
    const req = {
      body: {
        username: 'nonexistentuser',
        password: 'testpassword',
      },
    };
    const res = {
      status: sinon.stub().returnsThis(),
      json: sinon.stub(),
    };

    // Stub User.findOne to simulate user not found
    const findOneStub = sinon.stub(User, 'findOne').resolves(null);
    await authController.login(req, res);

    expect(res.status.calledWith(401)).to.be.true;
    expect(res.json.calledWith({ message: 'Invalid credentials' }))).to.be.true;

    // Restore the stubbed method
    findOneStub.restore();
  });
});

```

```

// ----- Test Case: 3 -----
it('should return an error if password is incorrect', async () => {
  const req = {
    body: {
      username: 'testuser',
      password: 'wrongpassword',
    },
  };
  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub(),
  };

  // Stub User.findOne to simulate finding the user
  const findOneStub = sinon.stub(User, 'findOne').resolves({
    _id: 'userId',
    username: 'testuser',
    password: 'hashedpassword',
    role: 'clerk',
  });

  // Stub bcrypt.compare to simulate password comparison
  const compareStub = sinon.stub(bcrypt, 'compare').resolves(false);

  await authController.login(req, res);

  expect(res.status.calledWith(401)).to.be.true;
  expect(res.json.calledWith({ message: 'Invalid credentials' })).to.be.true;

  // Restore the stubbed methods
  findOneStub.restore();
  compareStub.restore();
});

// ----- Test Case: 4 -----
it('should return an error if there is a server error', async () => {
  const req = {
    body: {
      username: 'testuser',
      password: 'testpassword',
    },
  };
  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub(),
  };

  // Stub User.findOne to simulate a server error
  const findOneStub = sinon.stub(User, 'findOne').throws(new Error('Server error'));

  await authController.login(req, res);

  expect(res.status.calledWith(500)).to.be.true;
  expect(res.json.calledWith(sinon.match({
    message: 'Error logging in',
    error: sinon.match.string,
  }))).to.be.true;

  // Restore the stubbed method
  findOneStub.restore();
});

```

vi. TCH – PIS – Main / services / auth-service / package.json

```
{
  "name": "auth-service",
  "version": "1.0.0",
  "main": "src/server.js",
  "scripts": {
    "test": "mocha test/**/*.test.js --exit",
    "start": "node src/server.js",
    "dev": "nodemon src/server.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "bcrypt": "^5.1.1",
    "bcryptjs": "^2.4.3",
    "body-parser": "^1.20.3",
    "dotenv": "^16.4.7",
    "express": "^4.21.2",
    "jsonwebtoken": "^9.0.2",
    "mongoose": "^8.9.5"
  },
  "description": "Authentication service for Patient Information System"
}
```

---

- b. patient-registration
- i. TCH – PIS – Main / services / patient-registration / src / controllers / patientController.js

```
const Patient = require('../models/Patient');

// Register a new patient (Only for clerks)
exports.registerPatient = async (req, res) => {
  try {
    const { firstName, lastName, mobile, email, diseaseHistory } = req.body;

    // Check if patient already exists
    const existingPatient = await Patient.findOne({ mobile });

    if (existingPatient) {
      return res.status(400).json({ message: 'Patient with this mobile number already exists' });
    }

    // Create and save new patient
    const newPatient = new Patient({ firstName, lastName, mobile, email, diseaseHistory });
    await newPatient.save();

    res.status(201).json({ message: 'Patient registered successfully', patientId: newPatient.patientId });
  } catch (error) {
    res.status(500).json({ message: 'Error registering patient', error: error.message });
  }
};

// Query all patients
exports.getAllPatients = async (req, res) => {
  try {
    const patients = await Patient.find(); // Fetch all patients from MongoDB
    res.status(200).json(patients);
  } catch (error) {
    res.status(500).json({ message: 'Error fetching patients', error: error.message });
  }
};

// Query a patient by objectId
exports.getPatientById = async (req, res) => {
  try {
    const { pid } = req.params;
    const patient = await Patient.findOne({ patientId: pid });

    if (!patient) {
      return res.status(404).json({ message: 'Patient not found' });
    }

    res.status(200).json(patient);
  } catch (error) {
    res.status(500).json({ message: 'Error fetching patient', error: error.message });
  }
};
```

ii. TCH – PIS – Main / services / patient-registration / src / middlewares / authMiddleware.js

```
const jwt = require('jsonwebtoken');

exports.verifyToken = (req, res, next) => {
  const token = req.header('Authorization');
  if (!token) return res.status(401).json({ message: 'Access Denied' });

  try {
    const verified = jwt.verify(token.replace('Bearer ', ''), process.env.JWT_SECRET);
    req.user = verified; // Attach user data to request
    next();
  } catch (error) {
    res.status(400).json({ message: 'Invalid Token' });
  }
};

exports.checkClerkRole = (req, res, next) => {
  if (req.user.role !== 'clerk') {
    return res.status(403).json({ message: 'Access Denied: Clerks only' });
  }
  next();
};
```

---

iii. TCH – PIS – Main / services / patient-registration / src / models / Patient.js

```
const mongoose = require('mongoose');
const Counter = require('../../shared/counterModel');
const patientSchema = new mongoose.Schema({
  patientId: {
    type: String,
    unique: true
  },
  firstName: {
    type: String,
    required: true
  },
  lastName: {
    type: String,
    required: true
  },
  mobile: {
    type: String,
    required: true,
    unique: true
  },
  email: {
    type: String,
    required: true,
  },
  diseaseHistory: {
    type: String,
  },
  createdAt: {
    type: Date,
    default: Date.now
  }
});
patientSchema.pre('save', async function (next) {
  if (!this.patientId) { // Only generate if not already set
    try {
      const counter = await Counter.findOneAndUpdate(
        { model: 'Patient' },
        { $inc: { count: 1 } },
        { new: true, upsert: true }
      );

      if (!counter || !counter.count) {
        return next(new Error("Counter update failed"));
      }

      this.patientId = counter.count;
    } catch (error) {
      return next(error);
    }
  }
  next();
});
module.exports = mongoose.model('Patient', patientSchema);
```

iv. TCH – PIS – Main / services / patient-registration / src / routes / patientRoutes.js

```
const express = require('express');
const router = express.Router();
const patientController = require('../controllers/patientController');
const { verifyToken, checkClerkRole } = require('../middlewares/authMiddleware');

router.post('/register', verifyToken, checkClerkRole, patientController.registerPatient);
router.get('/all', verifyToken, checkClerkRole, patientController.getAllPatients);
router.get('/:pid', verifyToken, checkClerkRole, patientController.getPatientById);

module.exports = router;
```

v. TCH – PIS – Main / services / patient-registration / src / server.js

```
const express = require('express');
const mongoose = require('mongoose');
const connectDB = require('../../shared/db');
const patientRoutes = require('../routes/patientRoutes');
require('dotenv').config({ path: '../../.env' });

const app = express();
```

```
// Middleware
app.use(express.json());

// Connect to MongoDB
connectDB();

// Routes
app.use('/patients', patientRoutes);

// Start the server
const PORT = 3001;
app.listen(PORT, () => console.log(` Patient service running on port ${PORT} `));
```

---

vi. TCH – PIS – Main / services / patient-registration / test / patient-registration.test.js

```
const chai = require('chai');
const sinon = require('sinon');
const Patient = require('../src/models/Patient');
const patientController = require('../src/controllers/patientController');
const expect = chai.expect;
describe('Patient Controller - Register - Unit - Test', () => {
  afterEach(() => {
    sinon.restore();
  });
});
```

```

// ----- Test Case: 1 -----
// Positive Test Case: Successful Patient Registration
it('should register a new patient successfully', async () => {
  const req = {
    body: {
      firstName: 'John',
      lastName: 'Doe',
      mobile: '1234567890',
      email: 'john.doe@example.com',
      diseaseHistory: 'None'
    }
  };

  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

  // Stub Patient.findOne to simulate no existing patient
  const findOneStub = sinon.stub(Patient, 'findOne').resolves(null);

  // Create a mock patient
  const mockPatient = new Patient(req.body);
  mockPatient.patientId = 'generated-id'; // Use a placeholder

  // Stub Patient.prototype.save to resolve with the mock patient
  const saveStub = sinon.stub(Patient.prototype, 'save').callsFake(function() {
    // Simulate the auto-increment behavior
    this.patientId = Math.floor(Math.random() * 10000).toString();
    return Promise.resolve(this);
  });

  // Call the controller method
  await patientController.registerPatient(req, res);

  // Assertions
  expect(res.status.calledWith(201)).to.be.true;

  const jsonCall = res.json.getCall(0);

  // Check the response structure
  expect(jsonCall.args[0]).to.be.an('object');
  expect(jsonCall.args[0]).to.have.property('message', 'Patient registered successfully');
  expect(jsonCall.args[0]).to.have.property('patientId');

  // Verify method calls
  expect(findOneStub.calledOnceWith({ mobile: req.body.mobile })).to.be.true;
  expect(saveStub.calledOnce).to.be.true;
});

```

```

// ----- Test Case: 2 -----
// Negative Test Case: Patient with Mobile Number Already Exists
it('should return an error if patient with mobile number already exists', async () => {
  const req = {
    body: {
      firstName: 'John',
      lastName: 'Doe',
      mobile: '1234567890',
      email: 'john.doe@example.com',
      diseaseHistory: 'None'
    }
  };

  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

  // Stub Patient.findOne to simulate existing patient
  sinon.stub(Patient, 'findOne').resolves({
    mobile: '1234567890'
  });

  await patientController.registerPatient(req, res);

  // Assertions
  expect(res.status.calledWith(400)).to.be.true;
  expect(res.json.calledWith({
    message: 'Patient with this mobile number already exists'
  })).to.be.true;
});

// ----- Test Case: 3 -----
// Negative Test Case: Server Error During Registration
it('should handle server errors during patient registration', async () => {
  const req = {
    body: {
      firstName: 'John',
      lastName: 'Doe',
      mobile: '1234567890',
      email: 'john.doe@example.com',
      diseaseHistory: 'None'
    }
  };

  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

  // Stub Patient.findOne to throw an error
  sinon.stub(Patient, 'findOne').throws(new Error('Database connection error'));

  await patientController.registerPatient(req, res);

  // Assertions
  expect(res.status.calledWith(500)).to.be.true;
  expect(res.json.calledWith(sinon.match({
    message: 'Error registering patient',
    error: sinon.match.string
  }))).to.be.true;
});

// Negative Test Case: Incomplete Patient Information
it('should return an error if required fields are missing', async () => {
  const req = {
    body: {
      // Intentionally leaving out some required fields
      firstName: '',
      lastName: '',
      mobile: '',
      email: ''
    }
  };

  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

```

```
};

// Stub Patient.findOne to simulate no existing patient
sinon.stub(Patient, 'findOne').resolves(null);

// Stub Patient.prototype.save to simulate save failure
sinon.stub(Patient.prototype, 'save').rejects(new Error('Validation failed'));

await patientController.registerPatient(req, res);

// Assertions
expect(res.status.calledWith(500)).to.be.true;
expect(res.json.calledWith(sinon.match({
  message: 'Error registering patient',
  error: sinon.match.string
}))).to.be.true;
});

});

describe('Patient Controller - Get All Patients - Unit - Test', () => {
  afterEach(() => {
    sinon.restore();
  });
});
```

```

//----- Test Case : 1 -----
// Positive Test Case: Successfully retrieve patients
it('should successfully retrieve all patients with complete details', async () => {
  const mockPatients = [
    {
      "_id": "67abada98e7056a95b8599ee",
      "firstName": "aa",
      "lastName": "aa",
      "mobile": "1212",
      "email": "aa@aa.aaa",
      "createdAt": "2025-02-11T20:06:01.811Z",
      "patientId": "1012",
      "__v": 0
    }
  ];

  const req = {};
  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

  // Stub Patient.find to return mock patients
  sinon.stub(Patient, 'find').resolves(mockPatients);

  await patientController.getAllPatients(req, res);

  // Assertions
  expect(res.status.calledWith(200)).to.be.true;

  // Verify the structure of the returned patient
  const returnedPatients = res.json.getCall(0).args[0];
  expect(returnedPatients).to.be.an('array');
  expect(returnedPatients[0]).to.have.all.keys(
    '_id',
    'firstName',
    'lastName',
    'mobile',
    'email',
    'createdAt',
    'patientId',
    '__v'
  );

  // Specific field validations
  expect(returnedPatients[0].patientId).to.equal('1012');
  expect(returnedPatients[0].mobile).to.equal('1212');
});

```

```

// ----- Test Case: 2 -----
// Negative Test Case: Error retrieving patients
it('should handle errors when fetching patients', async () => {
  const req = {};
  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

  // Stub Patient.find to throw an error
  sinon.stub(Patient, 'find').throws(new Error('Database connection error'));

  await patientController.getAllPatients(req, res);

  // Assertions
  expect(res.status.calledWith(500)).to.be.true;
  expect(res.json.calledWith(sinon.match({
    message: 'Error fetching patients',
    error: sinon.match.string
  }))).to.be.true;
});

// ----- Test Case: 3 -----
// Negative Test Case: No patients found (empty result)
it('should return an empty array when no patients exist', async () => {
  const req = {};
  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

  // Stub Patient.find to return an empty array
  sinon.stub(Patient, 'find').resolves([]);

  await patientController.getAllPatients(req, res);

  // Assertions
  expect(res.status.calledWith(200)).to.be.true;
  expect(res.json.calledWith([])).to.be.true;
});

```

```

// ----- Test Case: 4 -----
// Additional Test Case: Verify returned patients have correct fields
it('should validate returned patient fields', async () => {
  const mockPatients = [
    {
      _id: "67abada98e7056a95b8599ee",
      firstName: "aa",
      lastName: "aa",
      mobile: "1212",
      email: "aa@aa.aaa",
      createdAt: "2025-02-11T20:06:01.811Z",
      patientId: "1012",
      __v: 0
    }
  ];

  const req = {};
  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

  // Stub Patient.find to return mock patients
  sinon.stub(Patient, 'find').resolves(mockPatients);

  await patientController.getAllPatients(req, res);

  // Detailed field validations
  const returnedPatients = res.json.getCall(0).args[0];

  // Verify _id
  expect(returnedPatients[0]._id).to.be.a('string');
  expect(returnedPatients[0]._id).to.equal('67abada98e7056a95b8599ee');

  // Verify createdAt is a valid date
  expect(new Date(returnedPatients[0].createdAt).toString()).to.not.equal('Invalid Date');

  // Verify __v is a number
  expect(returnedPatients[0].__v).to.be.a('number');
  expect(returnedPatients[0].__v).to.equal(0);
});

describe('Patient Controller - Get Patient By ID - UNIT TEST', () => {
  afterEach(() => {
    sinon.restore();
  });
});

```

```

// ----- Test Case 1: -----
// Positive Test Case: Successfully retrieve patient by ID
it('should successfully retrieve a patient by patientId', async () => {
  const mockPatient = {
    "_id": "67abada98e7056a95b8599ee",
    "firstName": "John",
    "lastName": "Doe",
    "mobile": "1234567890",
    "email": "john.doe@example.com",
    "createdAt": "2025-02-11T20:06:01.811Z",
    "patientId": "1012",
    "__v": 0
  };

  const req = {
    params: {
      pid: '1012' // patientId to search
    }
  };

  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

  // Stub Patient.findOne to return mock patient
  sinon.stub(Patient, 'findOne').resolves(mockPatient);

  await patientController.getPatientById(req, res);

  // Assertions
  expect(Patient.findOne.calledOnceWith({ patientId: '1012' })).to.be.true;
  expect(res.status.calledWith(200)).to.be.true;

  // Verify returned patient details
  const returnedPatient = res.json.getCall(0).args[0];
  expect(returnedPatient).to.deep.equal(mockPatient);
  expect(returnedPatient.patientId).to.equal('1012');
});

```

```

// ----- Test Case:2 -----
// Negative Test Case: Patient Not Found
it('should return 404 when patient is not found', async () => {
  const req = {
    params: {
      pid: '9999' // Non-existent patient ID
    }
  };

  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

  // Stub Patient.findOne to return null
  sinon.stub(Patient, 'findOne').resolves(null);

  await patientController.getPatientById(req, res);

  // Assertions
  expect(Patient.findOne.calledOnceWith({ patientId: '9999' })).to.be.true;
  expect(res.status.calledWith(404)).to.be.true;
  expect(res.json.calledWith({ message: 'Patient not found' })).to.be.true;
});

// ----- Test Case: 3 -----
// Negative Test Case: Server Error
it('should handle server errors when fetching patient', async () => {
  const req = {
    params: {
      pid: '1012'
    }
  };

  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

  // Stub Patient.findOne to throw an error
  sinon.stub(Patient, 'findOne').throws(new Error('Database connection error'));

  await patientController.getPatientById(req, res);

  // Assertions
  expect(res.status.calledWith(500)).to.be.true;
  expect(res.json.calledWith(sinon.match({
    message: 'Error fetching patient',
    error: sinon.match.string
  }))).to.be.true;
});

// ----- Test Case: 4 -----
// Additional Test Case: Validate Patient Fields
it('should validate returned patient fields', async () => {
  const mockPatient = {
    "_id": "67abada98e7056a95b8599ee",
    "firstName": "John",
    "lastName": "Doe",
    "mobile": "1234567890",
    "email": "john.doe@example.com",
    "createdAt": "2025-02-11T20:06:01.811Z",
    "patientId": "1012",
    "__v": 0
  };

  const req = {
    params: {
      pid: '1012'
    }
  };

  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

  // Stub Patient.findOne to return mock patient

```

```
sinon.stub(Patient, 'findOne').resolves(mockPatient);

await patientController.getPatientById(req, res);

// Detailed field validations
const returnedPatient = res.json.getCall(0).args[0];

// Verify specific fields
expect(returnedPatient).to.have.all.keys(
  '_id',
  'firstName',
  'lastName',
  'mobile',
  'email',
  'createdAt',
  'patientId',
  '__v'
);

// Validate field types
expect(returnedPatient._id).to.be.a('string');
expect(returnedPatient.firstName).to.be.a('string');
expect(returnedPatient.mobile).to.be.a('string');
expect(new Date(returnedPatient.createdAt).toString()).to.not.equal('Invalid Date');
});
});
```

---

vii. TCH – PIS – Main / services / patient-registration / package.json

```
{
  "name": "patient-registration",
  "scripts": {
    "test": "mocha test/**/*.test.js --exit"
  },
  "devDependencies": {
    "mocha": "^11.1.0",
    "chai": "^5.1.0"
  }
}
```

---

c. Patient-treatment

i. TCH – PIS – Main / services / patient-treatment / src / controllers / treatmentController.js

```
const mongoose = require('mongoose');
const Treatment = require('../models/Treatment');
// Add a new diagnosis (Doctors Only)
exports.addDiagnosis = async (req, res) => {
  try {
    const { patientID, diagnosis, medications } = req.body;
    const doctorID = req.user.userId;
    if (!patientID || !diagnosis) {
      return res.status(400).json({ message: "Error: patientID and diagnosis are required" });
    }

    const objectIdPatientID = new mongoose.Types.ObjectId(patientID);
    let treatment = await Treatment.findOne({ patientID: objectIdPatientID, diagnosis });

    if (treatment) {
      console.log("🔧 Updating existing diagnosis...");
      treatment.medications = medications;
    } else {
      treatment = new Treatment({ patientID: objectIdPatientID, doctorID, diagnosis, medications });
    }

    const savedTreatment = await treatment.save();

    res.status(201).json({ message: 'Diagnosis recorded successfully', treatment: savedTreatment });
  } catch (error) {
    res.status(500).json({ message: 'Error recording diagnosis', error: error.message });
  }
};

// Update Medications Separately (Doctors Only)
exports.updateMedications = async (req, res) => {
  try {
    const { patientID } = req.params;
    const { medications } = req.body;
    if (!medications || !Array.isArray(medications)) {
      return res.status(400).json({ message: "Error: Medications must be an array" });
    }

    const objectIdPatientID = new mongoose.Types.ObjectId(patientID);
    let treatment = await Treatment.findOne({ patientID: objectIdPatientID });
    if (!treatment) {
      return res.status(404).json({ message: "No treatment record found for this patient" });
    }
    treatment.medications = medications;
    await treatment.save();
    res.status(200).json({ message: "Medications updated successfully", treatment });
  } catch (error) {
    res.status(500).json({ message: 'Error updating medications', error: error.message });
  }
};

// Remove a Specific Medication (Doctors Only)
exports.removeMedication = async (req, res) => {
  try {
    const { patientID, medication } = req.params;
    const objectIdPatientID = new mongoose.Types.ObjectId(patientID);
    let treatment = await Treatment.findOne({ patientID: objectIdPatientID });
    if (!treatment) {
      return res.status(404).json({ message: "No treatment record found for this patient" });
    }

    // 🐞 Debug: Show comparison between stored and requested medication
    treatment.medications.forEach((med) => {
      // Use case-insensitive, whitespace-trimmed comparison
      const updatedMedications = treatment.medications.filter(
        med => med.trim().toLowerCase() !== medication.trim().toLowerCase()
      );
      if (updatedMedications.length === treatment.medications.length) {
        return res.status(400).json({ message: "Medication not found in treatment record" });
      }
      treatment.medications = updatedMedications;
      await treatment.save();
      res.status(200).json({ message: "Medication removed successfully", treatment });
    } catch (error) {

```

```

    res.status(500).json({ message: 'Error removing medication', error: error.message });
  }
};

// Retrieve Medications Only (Doctors & Nurses)
exports.getMedications = async (req, res) => {
  try {
    const { patientID } = req.params;
    const objectIdPatientID = new mongoose.Types.ObjectId(patientID);
    let treatment = await Treatment.findOne({ patientID: objectIdPatientID });
    if (!treatment || !treatment.medications.length) {
      return res.status(404).json({ message: "No medications found for this patient" });
    }
    res.status(200).json({ medications: treatment.medications });
  } catch (error) {
    res.status(500).json({ message: 'Error retrieving medications', error: error.message });
  }
};

// Log patient vitals (Nurses Only)
exports.addVitals = async (req, res) => {
  try {
    const { patientID, temperature, bloodPressure } = req.body;
    const update = { temperature, bloodPressure, time: new Date() };
    if (!patientID || !temperature || !bloodPressure) {
      return res.status(400).json({ message: "Error: patientID, temperature, and bloodPressure are required" });
    }
    const objectIdPatientID = new mongoose.Types.ObjectId(patientID);
    let treatment = await Treatment.findOne({ patientID: objectIdPatientID });
    if (!treatment) {
      treatment = new Treatment({ patientID: objectIdPatientID, vitals: [update] });
    } else {
      treatment.vitals.push(update);
    }
    const savedTreatment = await treatment.save();

    res.status(201).json({ message: 'Vitals recorded successfully', treatment: savedTreatment });
  } catch (error) {
    res.status(500).json({ message: 'Error logging vitals', error: error.message });
  }
};

// Retrieve a patient's treatment history
exports.getTreatmentByPatientID = async (req, res) => {
  try {
    const { patientID } = req.params;
    const objectIdPatientID = new mongoose.Types.ObjectId(patientID);
    const treatment = await Treatment.find({ patientID: objectIdPatientID });
    if (!treatment.length) {
      return res.status(404).json({ message: 'No treatment records found for this patient' });
    }
    res.status(200).json({ treatment });
  } catch (error) {
    res.status(500).json({ message: 'Error fetching treatment history', error: error.message });
  }
};

```

ii. TCH – PIS – Main / services / patient-treatment / src / middleware / authMiddleware.js

```
const jwt = require('jsonwebtoken');
require('dotenv').config({ path: '../.env' }); // Load shared environment variables

// Middleware to verify JWT token
exports.verifyToken = (req, res, next) => {
  const token = req.header('Authorization'); // Get token from request headers
  if (!token) return res.status(401).json({ message: 'Access Denied: No Token Provided' });

  try {
    const verified = jwt.verify(token.replace('Bearer ', ''), process.env.JWT_SECRET); // Verify JWT
    req.user = verified; // Attach user data (userId, role) to request
    next(); // Proceed to the next function
  } catch (error) {
    res.status(400).json({ message: 'Invalid Token' });
  }
};

// Middleware to allow only doctors
exports.checkDoctorRole = (req, res, next) => {
  if (req.user.role !== 'doctor') {
    return res.status(403).json({ message: 'Access Denied: Doctors only' });
  }
  next();
};

// Middleware to allow only nurses
exports.checkNurseRole = (req, res, next) => {
  if (req.user.role !== 'nurse') {
    return res.status(403).json({ message: 'Access Denied: Nurses only' });
  }
  next();
};
```

---

iii. TCH – PIS – Main / services / patient-treatment / src / models / Treatment.js

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

const treatmentSchema = new mongoose.Schema({
  patientID: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'Patient',
    required: true
  },
  doctorID: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User',
    required: false
  },
  diagnosis: {
    type: String,
    required: false
  },
  medications: {
    type: [String]
  },
  vitals: [{
    temperature: Number,
    bloodPressure: String,
    time: { type: Date, default: Date.now }
  }],
  createdAt: {
    type: Date,
    default: Date.now
  }
});
module.exports = mongoose.model('Treatment', treatmentSchema);
```

iv. TCH – PIS – Main / services / patient-treatment / src / routes / treatmentRoutes.js

```
const express = require('express');
const router = express.Router();
const treatmentController = require('../controllers/treatmentController');
const { verifyToken, checkDoctorRole, checkNurseRole } = require('../middleware/authMiddleware');

// Only doctors can add a diagnosis
router.post('/treatment/diagnosis', verifyToken, checkDoctorRole, treatmentController.addDiagnosis);

// Only nurses can log patient vitals
router.post('/treatment/vitals', verifyToken, checkNurseRole, treatmentController.addVitals);

// Retrieve a patient's treatment history
router.get('/treatment/:patientID', verifyToken, treatmentController.getTreatmentByPatientID);

// New Route: Update medications separately (Doctors Only)
router.put('/treatment/medications/:patientID', verifyToken, checkDoctorRole, treatmentController.updateMedications);

// New Route: Remove a specific medication (Doctors Only)
router.delete('/treatment/medications/:patientID/:medication', verifyToken, checkDoctorRole,
treatmentController.removeMedication);

// New Route: Get only medications for a patient (Doctors & Nurses)
router.get('/treatment/medications/:patientID', verifyToken, treatmentController.getMedications);

module.exports = router;
```

v. TCH – PIS – Main / services / patient-treatment / src / server.js

```
const express = require('express');
const mongoose = require('mongoose');
const connectDB = require('../shared/db');
const treatmentRoutes = require('./routes/treatmentRoutes');
// require('dotenv').config({ path: '../.env' });
// require('dotenv').config({ path: '../.env' });
require('dotenv').config({ path: __dirname + '/../.env' });

// console.log(` 🐞 DEBUG: Loaded PORT from .env: ${process.env.PORT}`);
// console.log(` 🐞 DEBUG: Loaded Treatment Service Port: ${process.env.TREATMENT_SERVICE_PORT}`);

const app = express();

// Middleware
app.use(express.json());

// Connect to MongoDB
connectDB();

// Mount Treatment Routes
app.use('/api', treatmentRoutes);

// Debug: Log Registered Routes
// const listRoutes = (app) => {
//   console.log("\n 🐞 Registered Routes:");
//   app._router.stack.forEach((middleware) => {
//     if (middleware.route) {
//       console.log(` ${Object.keys(middleware.route.methods).join(', ').toUpperCase()} ${middleware.route.path}`);
//     } else if (middleware.name === 'router') {
//       middleware.handle.stack.forEach((handler) => {
//         if (handler.route) {
//           console.log(` ${Object.keys(handler.route.methods).join(', ').toUpperCase()} /api${handler.route.path}`);
//         }
//       });
//     }
//   });
// };

// Start the server
// const PORT = process.env.PORT || 3002;
// app.listen(PORT, () => {
//   console.log(` Patient Treatment Service running on port ${PORT}`);
//   listRoutes(app); // Log routes after server starts
// });

const PORT = process.env.SERVICE_NAME === "treatment" ? 3002 : process.env.PORT || 3002;

app.listen(PORT, () => {
  console.log(` Patient Treatment Service running on port ${PORT}`);
});

// const express = require('express');
// const mongoose = require('mongoose');
// const connectDB = require('../shared/db');
// const treatmentRoutes = require('./routes/treatmentRoutes');
// require('dotenv').config({ path: '../.env' });

// const app = express();

// // Middleware
// app.use(express.json());

// // Connect to MongoDB
// connectDB();

// // Mount Treatment Routes
// app.use('/api', treatmentRoutes);

// // Start the server
// const PORT = process.env.PORT || 3002;
// app.listen(PORT, () => console.log(` Patient Treatment Service running on port ${PORT}`));
```

```

const chai = require('chai');
const sinon = require('sinon');
const mongoose = require('mongoose');
const Treatment = require('../src/models/Treatment');
const treatmentController = require('../src/controllers/treatmentController');
const { expect } = chai;

describe('Treatment Controller - addDiagnosis - Unit - Test', () => {
  afterEach(() => {
    sinon.restore();
  });

  //----- Test Case: 1 -----
  //Positive Test: Successfully create a new diagnosis

  it('should create a new diagnosis successfully', async () => {
    // Arrange
    const req = {
      body: {
        patientID: '67abada98e7056a95b8599ee',
        diagnosis: 'Sample diagnosis',
        medications: ['Med1', 'Med2']
      },
      user: { userId: '67bb267e763edeab0e1753f5' }
    };
    const res = {
      status: sinon.stub().returnsThis(),
      json: sinon.stub()
    };

    // Stub Treatment.findOne to simulate "no existing diagnosis"
    sinon.stub(Treatment, 'findOne').resolves(null);

    // Create a mock Treatment instance that returns the same values as in the controller
    const mockTreatment = {
      _id: new mongoose.Types.ObjectId(),
      patientID: req.body.patientID,
      doctorID: req.user.userId,
      diagnosis: req.body.diagnosis,
      medications: req.body.medications,
      save: sinon.stub().resolvesThis() // allow .save() to resolve with the same object
    };

    sinon.stub(Treatment, 'constructor').returns(mockTreatment);
    sinon.stub(Treatment.prototype, 'save').resolves(mockTreatment);

    await treatmentController.addDiagnosis(req, res);

    // Assert
    expect(res.status.calledOnceWithExactly(201)).to.be.true;

    // Grab the actual response argument
    const responseData = res.json.getCall(0).args[0];
    expect(responseData).to.have.property('message', 'Diagnosis recorded successfully');
    expect(responseData).to.have.property('treatment');
    expect(responseData.treatment).to.include({
      patientID: req.body.patientID,
      diagnosis: req.body.diagnosis
    });
    // Check array content for medications if desired
    expect(responseData.treatment.medications).to.deep.equal(req.body.medications);
  });
}

```

```

// ----- Test Case: 2 -----
it('should return 400 if patientID or diagnosis is missing', async () => {

  const req = {
    body: {
      // patientID is missing
      diagnosis: 'Some diagnosis'
    },
    user: { userId: '67bb267e763edeab0e1753f5' }
  };
  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

  await treatmentController.addDiagnosis(req, res);

  expect(res.status.calledOnceWithExactly(400)).to.be.true;
  expect(res.json.calledWithMatch({
    message: 'Error: patientID and diagnosis are required'
  })).to.be.true;
});

// ----- Test Case: 3 -----
it('should return 500 if an exception is thrown', async () => {
  // Arrange
  const req = {
    body: {
      patientID: '67abada98e7056a95b8599ee',
      diagnosis: 'Throw error test'
    },
    user: { userId: '67bb267e763edeab0e1753f5' }
  };
  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

  // Force an error when calling Treatment.findOne
  sinon.stub(Treatment, 'findOne').throws(new Error('DB error'));

  // Act
  await treatmentController.addDiagnosis(req, res);

  // Assert
  expect(res.status.calledOnceWithExactly(500)).to.be.true;
  expect(res.json.calledWithMatch({
    message: 'Error recording diagnosis',
    error: 'DB error'
  })).to.be.true;
});

describe('Treatment Controller - updateMedications - UNIT - TEST', () => {
  afterEach(() => {
    sinon.restore(); // Reset all stubs after each test
  });
});

```

```
// ----- Test Case: 1 -----
it('should update medications successfully', async () => {
  // Arrange
  const req = {
    params: { patientID: '67abada98e7056a95b8599ee' },
    body: { medications: ['MedA', 'MedB'] }
  };
  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

  // Stub Treatment.findOne to simulate a found record
  const mockTreatment = {
    _id: new mongoose.Types.ObjectId(),
    patientID: req.params.patientID,
    medications: ['OldMed1'],
    save: sinon.stub().resolvesThis() // Simulate successful save()
  };
  sinon.stub(Treatment, 'findOne').resolves(mockTreatment);

  // Act
  await treatmentController.updateMedications(req, res);

  // Assert
  expect(res.status.calledOnceWithExactly(200)).to.be.true;
  expect(res.json.calledWithMatch({
    message: 'Medications updated successfully',
    treatment: { patientID: req.params.patientID, medications: req.body.medications }
  })).to.be.true;
  expect(mockTreatment.medications).to.deep.equal(req.body.medications); // Ensure medications are updated
});

// ----- Test Case: 2 -----
// Negative Test: Returns 400 if medications field is missing or not an array
it('should return 400 if medications field is missing', async () => {
  // Arrange
  const req = {
    params: { patientID: '67abada98e7056a95b8599ee' },
    body: {} // No medications field
  };
  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

  // Act
  await treatmentController.updateMedications(req, res);

  // Assert
  expect(res.status.calledOnceWithExactly(400)).to.be.true;
  expect(res.json.calledWithMatch({
    message: 'Error: Medications must be an array'
  })).to.be.true;
});

// ----- Test Case: 3 -----
it('should return 400 if medications is not an array', async () => {
  // Arrange
  const req = {
    params: { patientID: '67abada98e7056a95b8599ee' },
    body: { medications: 'Not an array' } // Invalid data type
  };
  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

  // Act
  await treatmentController.updateMedications(req, res);

  // Assert
  expect(res.status.calledOnceWithExactly(400)).to.be.true;
  expect(res.json.calledWithMatch({
    message: 'Error: Medications must be an array'
  })).to.be.true;
});
```

```

// ----- Test Case:4 -----
// Negative Test: Returns 404 if no treatment record is found
it('should return 404 if no treatment record is found', async () => {
  // Arrange
  const req = {
    params: { patientID: '67abada98e7056a95b8599ee' },
    body: { medications: ['MedX', 'MedY'] }
  };
  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

  // Stub Treatment.findOne to return null (patient not found)
  sinon.stub(Treatment, 'findOne').resolves(null);

  // Act
  await treatmentController.updateMedications(req, res);

  // Assert
  expect(res.status.calledOnceWithExactly(404)).to.be.true;
  expect(res.json.calledWithMatch({
    message: 'No treatment record found for this patient'
  })).to.be.true;
});

// ----- Test Case:5 -----
// Negative Test: Returns 500 if a database error occurs
it('should return 500 if an internal error occurs', async () => {
  // Arrange
  const req = {
    params: { patientID: '67abada98e7056a95b8599ee' },
    body: { medications: ['MedX', 'MedY'] }
  };
  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

  // Force an error when calling Treatment.findOne
  sinon.stub(Treatment, 'findOne').throws(new Error('Database failure'));

  // Act
  await treatmentController.updateMedications(req, res);

  // Assert
  expect(res.status.calledOnceWithExactly(500)).to.be.true;
  expect(res.json.calledWithMatch({
    message: 'Error updating medications',
    error: 'Database failure'
  })).to.be.true;
});

describe('Treatment Controller - removeMedication - UNIT - TEST', () => {
  afterEach(() => {
    sinon.restore(); // Reset all stubs after each test
  });

  // ----- Test Case: 1 -----
  // Positive Test: Successfully remove a medication
  it('should remove a medication successfully', async () => {
    // Arrange
    const req = {
      params: { patientID: '67abada98e7056a95b8599ee', medication: 'MedA' }
    };
    const res = {
      status: sinon.stub().returnsThis(),
      json: sinon.stub()
    };

    // Stub Treatment.findOne to simulate a found record
    const mockTreatment = {
      _id: new mongoose.Types.ObjectId(),
      patientID: req.params.patientID,
      medications: ['MedA', 'MedB'],
      save: sinon.stub().resolvesThis() // Simulate successful save()
    };
    sinon.stub(Treatment, 'findOne').resolves(mockTreatment);
  });

```

```

// Act
await treatmentController.removeMedication(req, res);

// Assert
expect(res.status.calledOnceWithExactly(200)).to.be.true;
expect(res.json.calledWithMatch({
  message: 'Medication removed successfully',
  treatment: { patientID: req.params.patientID }
})).to.be.true;
expect(mockTreatment.medications).to.deep.equal(['MedB']); // Ensure MedA was removed
});

// ----- Test Case 2: -----
// Negative Test: Returns 404 if no treatment record is found
it('should return 404 if no treatment record is found', async () => {
  // Arrange
  const req = {
    params: { patientID: '67abada98e7056a95b8599ee', medication: 'MedA' }
  };
  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

  // Stub Treatment.findOne to return null (patient not found)
  sinon.stub(Treatment, 'findOne').resolves(null);

  // Act
  await treatmentController.removeMedication(req, res);

  // Assert
  expect(res.status.calledOnceWithExactly(404)).to.be.true;
  expect(res.json.calledWithMatch({
    message: 'No treatment record found for this patient'
  })).to.be.true;
});

// ----- Test Case: 3 -----
// Negative Test: Returns 400 if the medication is not found in the treatment record
it('should return 400 if the medication does not exist in the treatment record', async () => {
  // Arrange
  const req = {
    params: { patientID: '67abada98e7056a95b8599ee', medication: 'MedX' }
  };
  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

  // Stub Treatment.findOne to simulate an existing patient but without the requested medication
  const mockTreatment = {
    _id: new mongoose.Types.ObjectId(),
    patientID: req.params.patientID,
    medications: ['MedA', 'MedB'], // MedX is not present
    save: sinon.stub().resolvesThis()
  };
  sinon.stub(Treatment, 'findOne').resolves(mockTreatment);

  // Act
  await treatmentController.removeMedication(req, res);

  // Assert
  expect(res.status.calledOnceWithExactly(400)).to.be.true;
  expect(res.json.calledWithMatch({
    message: 'Medication not found in treatment record'
  })).to.be.true;
});

```

```

// ----- Test Case: 4 -----
// Negative Test: Returns 500 if an internal error occurs
it('should return 500 if an internal error occurs', async () => {
  // Arrange
  const req = {
    params: { patientID: '67abada98e7056a95b8599ee', medication: 'MedA' }
  };
  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

  // Force an error when calling Treatment.findOne
  sinon.stub(Treatment, 'findOne').throws(new Error('Database failure'));

  // Act
  await treatmentController.removeMedication(req, res);

  // Assert
  expect(res.status.calledOnceWithExactly(500)).to.be.true;
  expect(res.json.calledWithMatch({
    message: 'Error removing medication',
    error: 'Database failure'
  })).to.be.true;
});

});

describe('Treatment Controller - getMedications - UNIT - TEST ', () => {
  afterEach(() => {
    sinon.restore(); // Reset all stubs after each test
  });

  // ----- Test Case: 1 -----
  // Positive Test: Successfully retrieves medications
  it('should retrieve medications successfully', async () => {
    // Arrange
    const req = { params: { patientID: '67abada98e7056a95b8599ee' } };
    const res = {
      status: sinon.stub().returnsThis(),
      json: sinon.stub()
    };

    // Stub Treatment.findOne to simulate a found record with medications
    const mockTreatment = {
      _id: new mongoose.Types.ObjectId(),
      patientID: req.params.patientID,
      medications: ['MedA', 'MedB']
    };
    sinon.stub(Treatment, 'findOne').resolves(mockTreatment);

    // Act
    await treatmentController.getMedications(req, res);

    // Assert
    expect(res.status.calledOnceWithExactly(200)).to.be.true;
    expect(res.json.calledWithMatch({ medications: ['MedA', 'MedB'] })).to.be.true;
  });
});

```

```

// ----- Test Case: 2 -----
// Negative Test: Returns 404 if no treatment record is found
it('should return 404 if no treatment record is found', async () => {
  // Arrange
  const req = { params: { patientID: '67abada98e7056a95b8599ee' } };
  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

  // Stub Treatment.findOne to return null (patient not found)
  sinon.stub(Treatment, 'findOne').resolves(null);

  // Act
  await treatmentController.getMedications(req, res);

  // Assert
  expect(res.status.calledOnceWithExactly(404)).to.be.true;
  expect(res.json.calledWithMatch({
    message: 'No medications found for this patient'
  })).to.be.true;
});

// ----- Test Case: 3 -----
// Negative Test: Returns 404 if treatment exists but has no medication
it('should return 404 if the treatment record exists but has no medications', async () => {
  // Arrange
  const req = { params: { patientID: '67abada98e7056a95b8599ee' } };
  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

  // Stub Treatment.findOne to simulate an existing treatment record with empty medications
  const mockTreatment = {
    _id: new mongoose.Types.ObjectId(),
    patientID: req.params.patientID,
    medications: []
  };
  sinon.stub(Treatment, 'findOne').resolves(mockTreatment);

  // Act
  await treatmentController.getMedications(req, res);

  // Assert
  expect(res.status.calledOnceWithExactly(404)).to.be.true;
  expect(res.json.calledWithMatch({
    message: 'No medications found for this patient'
  })).to.be.true;
});

// ----- Test Case: 4 -----
// Negative Test: Returns 500 if an internal error occurs
it('should return 500 if an internal error occurs', async () => {
  // Arrange
  const req = { params: { patientID: '67abada98e7056a95b8599ee' } };
  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

  // Force an error when calling Treatment.findOne
  sinon.stub(Treatment, 'findOne').throws(new Error('Database failure'));

  // Act
  await treatmentController.getMedications(req, res);

  // Assert
  expect(res.status.calledOnceWithExactly(500)).to.be.true;
  expect(res.json.calledWithMatch({
    message: 'Error retrieving medications',
    error: 'Database failure'
  })).to.be.true;
});

describe('Treatment Controller - addVitals - UNIT - TEST', () => {
  afterEach(() => {
    sinon.restore(); // Reset all stubs after each test
  });
});

```

```

// ----- Test Case:1 -----
// Positive Test: Logs vitals successfully for an existing patient record
it('should log vitals for an existing treatment record', async () => {
  // Arrange
  const req = {
    body: {
      patientID: '67abada98e7056a95b8599ee',
      temperature: 98.6,
      bloodPressure: '120/80'
    }
  };
  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

  // Stub Treatment.findOne to return an existing record
  const mockTreatment = {
    _id: new mongoose.Types.ObjectId(),
    patientID: req.body.patientID,
    vitals: [],
    save: sinon.stub().resolvesThis() // Mock save method
  };
  sinon.stub(Treatment, 'findOne').resolves(mockTreatment);

  // Act
  await treatmentController.addVitals(req, res);

  // Assert
  expect(mockTreatment.vitals).toHaveLength(1); // Ensure vitals were added
  expect(res.status.calledOnceWithExactly(201)).toBe(true);
  expect(res.json.calledWithMatch({
    message: 'Vitals recorded successfully'
  })).toBe(true;
});

// ----- Test case: 2 -----
// Positive Test: Creates a new treatment record if no existing record is found
it('should create a new treatment record if patient has no existing record', async () => {
  // Arrange
  const req = {
    body: {
      patientID: '67abada98e7056a95b8599ee',
      temperature: 99.1,
      bloodPressure: '130/85'
    }
  };
  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

  // Stub Treatment.findOne to return null (no existing record)
  sinon.stub(Treatment, 'findOne').resolves(null);

  // Stub Treatment.prototype.save to simulate saving a new record
  const mockNewTreatment = new Treatment({
    patientID: req.body.patientID,
    vitals: [{ temperature: req.body.temperature, bloodPressure: req.body.bloodPressure, time: new Date() }]
  });
  sinon.stub(Treatment.prototype, 'save').resolves(mockNewTreatment);

  // Act
  await treatmentController.addVitals(req, res);

  // Assert
  expect(res.status.calledOnceWithExactly(201)).toBe(true);
  expect(res.json.calledWithMatch({
    message: 'Vitals recorded successfully'
  })).toBe(true;
});

// ----- Test Case 3: -----
// Negative Test: Returns 400 if required fields are missing
it('should return 400 if required fields are missing', async () => {
  // Arrange
  const req = { body: { patientID: '67abada98e7056a95b8599ee' } }; // Missing temperature & bloodPressure
  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

```

```

// Act
await treatmentController.addVitals(req, res);

// Assert
expect(res.status.calledOnceWithExactly(400)).to.be.true;
expect(res.json.calledWithMatch({
  message: 'Error: patientID, temperature, and bloodPressure are required'
})).to.be.true;
});

// ----- Test Case:4 -----
it('should return 500 if an internal server error occurs', async () => {
  // Arrange
  const req = {
    body: {
      patientID: '67abada98e7056a95b8599ee',
      temperature: 97.5,
      bloodPressure: '110/75'
    }
  };
  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

  // Stub Treatment.findOne to throw an error
  sinon.stub(Treatment, 'findOne').throws(new Error('Database failure'));

  // Act
  await treatmentController.addVitals(req, res);

  // Assert
  expect(res.status.calledOnceWithExactly(500)).to.be.true;
  expect(res.json.calledWithMatch({
    message: 'Error logging vitals',
    error: 'Database failure'
})).to.be.true;
});

});

describe(' Treatment Controller - getTreatmentByPatientID - UNIT - TEST ', () => {
  afterEach(() => {
    sinon.restore(); // Reset all stubs after each test
  });

  // ----- Test Case:1 -----
  // Positive Test: Successfully retrieves treatment history for a valid patient ID
  it('should return treatment history for a valid patient ID', async () => {
    // Arrange
    const req = { params: { patientID: '67abada98e7056a95b8599ee' } };
    const res = {
      status: sinon.stub().returnsThis(),
      json: sinon.stub()
    };

    // Mock treatment data
    const mockTreatment = [
      { patientID: req.params.patientID, medications: ['Paracetamol'], vitals: [{ temperature: 98.6, bloodPressure: '120/80' }] }
    ];

    // Stub Treatment.find to return treatment history
    sinon.stub(Treatment, 'find').resolves(mockTreatment);

    // Act
    await treatmentController.getTreatmentByPatientID(req, res);

    // Assert
    expect(res.status.calledOnceWithExactly(200)).to.be.true;
    expect(res.json.calledWithMatch({ treatment: mockTreatment })).to.be.true;
  });
});

```

```

// ----- Test Case: 2 -----
// Positive Test: Handles case where no treatment history is found (returns 404)
it('should return 404 if no treatment records are found for the patient', async () => {
  // Arrange
  const req = { params: { patientID: '67abada98e7056a95b8599ff' } };
  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

  // Stub Treatment.find to return an empty array
  sinon.stub(Treatment, 'find').resolves([]);

  // Act
  await treatmentController.getTreatmentByPatientID(req, res);

  // Assert
  expect(res.status.calledOnceWithExactly(404)).to.be.true;
  expect(res.json.calledWithMatch({ message: 'No treatment records found for this patient' })).to.be.true;
});

// ----- Test Case: 3 -----
// Negative Test: Returns 400 if patientID is missing or invalid
it('should return 404 if patientID is invalid or not found', async () => {
  // Arrange
  const req = { params: { patientID: 9999999 } }; // Invalid patient ID
  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

  // Stub Treatment.find to return an empty array (simulating no records found)
  sinon.stub(Treatment, 'find').resolves([]);

  // Act
  await treatmentController.getTreatmentByPatientID(req, res);

  // Assert
  expect(res.status.calledOnceWithExactly(404)).to.be.true;
  expect(res.json.calledWithMatch({ message: 'No treatment records found for this patient' })).to.be.true;
});

// ----- Test Case: 5 -----
// Negative Test: Returns 500 if an internal server error occur
it('should return 500 if a database error occurs', async () => {
  // Arrange
  const req = { params: { patientID: '67abada98e7056a95b8599ee' } };
  const res = {
    status: sinon.stub().returnsThis(),
    json: sinon.stub()
  };

  // Stub Treatment.find to throw an error
  sinon.stub(Treatment, 'find').throws(new Error('Database failure'));

  // Act
  await treatmentController.getTreatmentByPatientID(req, res);

  // Assert
  expect(res.status.calledOnceWithExactly(500)).to.be.true;
  expect(res.json.calledWithMatch({
    message: 'Error fetching treatment history',
    error: 'Database failure'
  })).to.be.true;
});
});

```

vii. TCH – PIS – Main / services / patient-treatment / .env

```
MONGO_URI=mongodb+srv://fmc4000:PcMk4CYYjrNQ9LW8@patient-info-  
system.qj5ku.mongodb.net/patientdb?retryWrites=true&w=majority  
JWT_SECRET=e5b8a6d8f92c  
PORT=3002
```

---

viii. TCH – PIS – Main / services / patient-treatment / package.json

```
{  
  "name": "patient-treatment",  
  "version": "1.0.0",  
  "main": "index.js",  
  "scripts": {  
    "test": "mocha test/**/*.test.js --exit"  
  },  
  "keywords": [],  
  "author": "",  
  "license": "ISC",  
  "description": "",  
  "dependencies": {  
    "axios": "^1.7.9",  
    "cors": "^2.8.5",  
    "dotenv": "^16.4.7",  
    "express": "^4.21.2",  
    "jsonwebtoken": "^9.0.2",  
    "mongoose": "^8.9.6"  
  }  
}
```

---