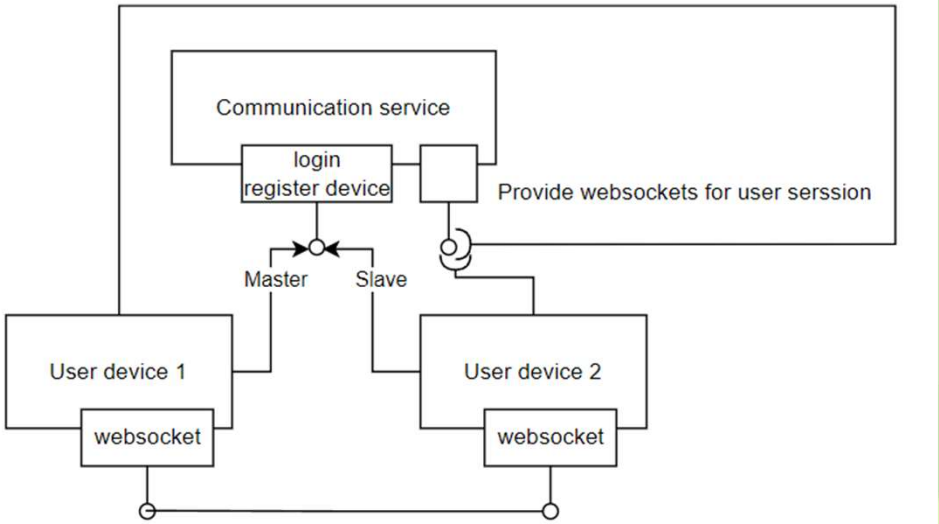
Communication service

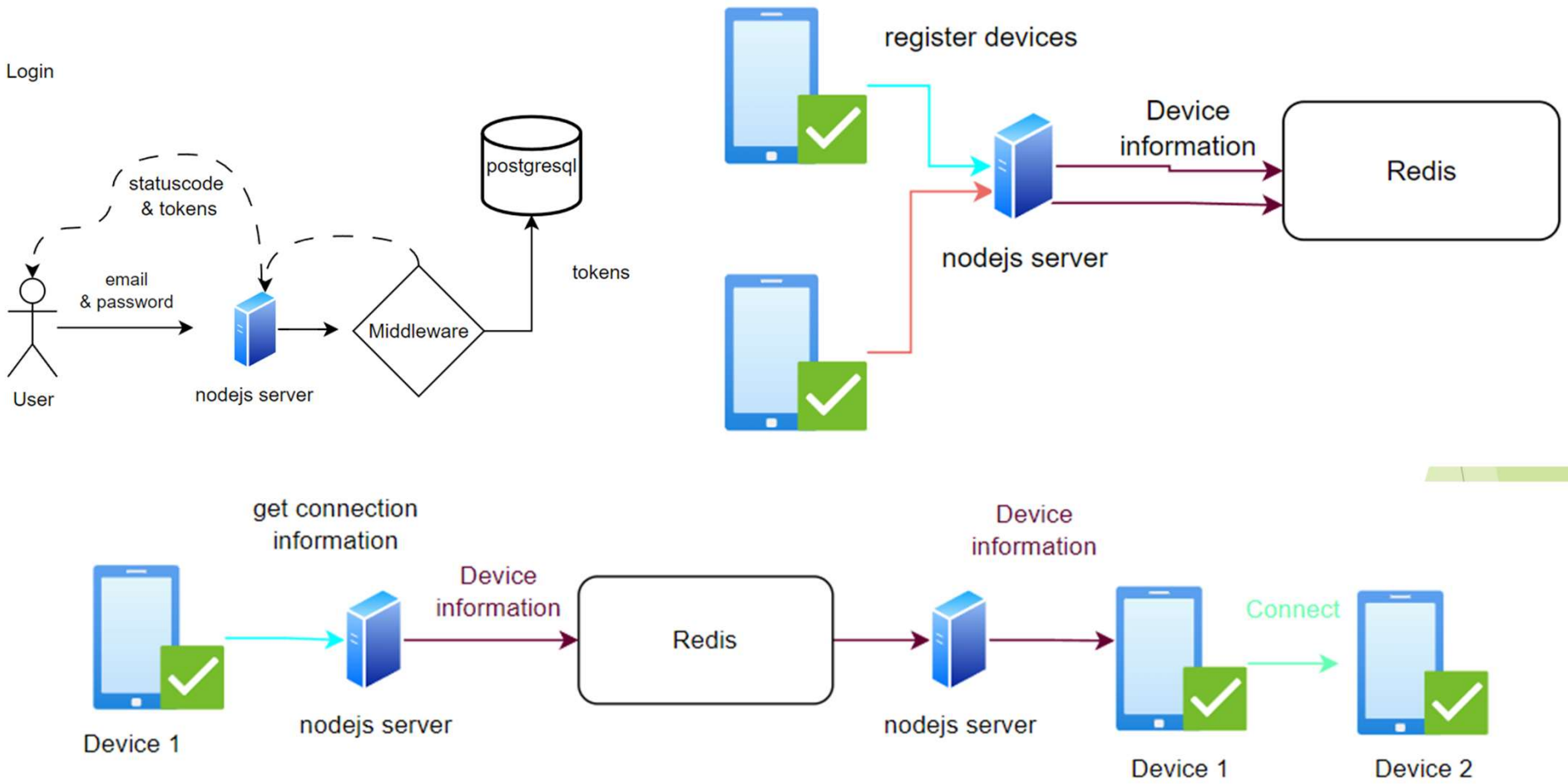
Secure programming coursework

Communication service for pairing devices

- Mobile devices usually have dynamic ip-address and available ports might vary
- This platform enables connecting devices easily and securely.

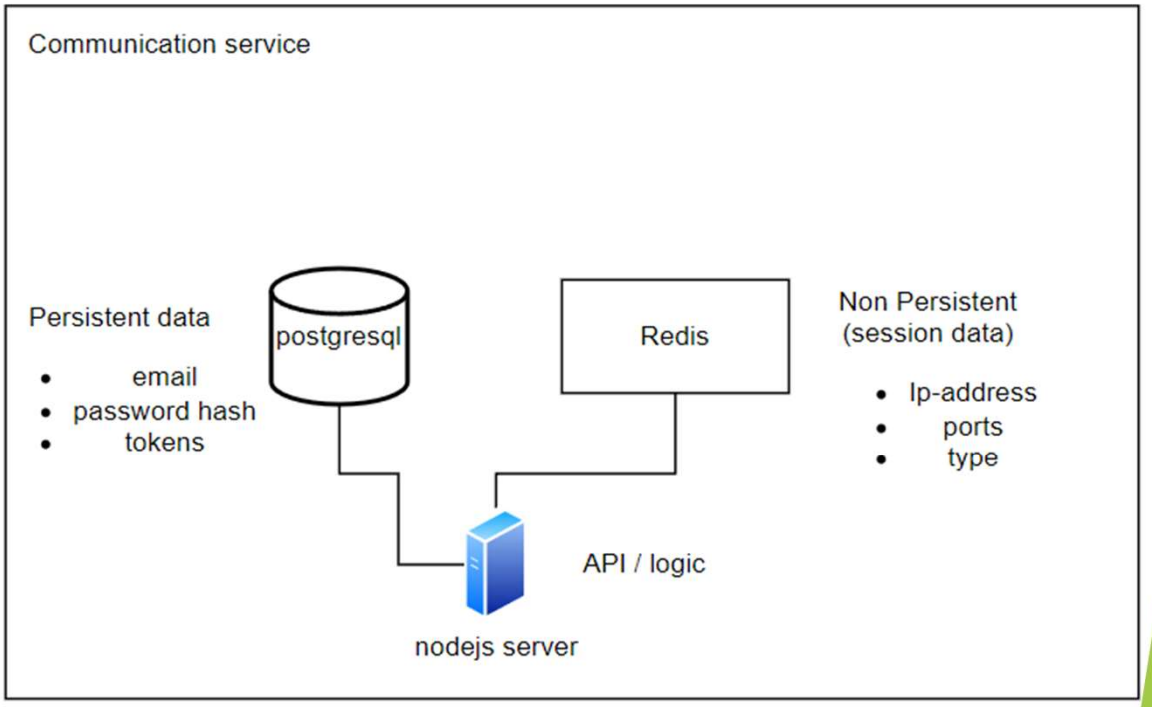


How it works


Login
statuscode
& tokens
email
& password
User
nodejs server
Middleware
postgresql
tokens
register devices
Device
information
Redis
nodejs server
get connection
information
Device 1
nodejs server
Device
information
Redis
Device
information
nodejs server
Connect
Device 1
Device 2

Structure of the program

```
├── README.md
├── backend
│   ├── Dockerfile
│   ├── Dockerfile.tests
│   ├── package.json
│   ├── server.js
│   ├── sonar-project.properties
│   └── src
│       ├── config
│       │   └── db.js
│       ├── controllers
│       │   ├── authController.js
│       │   ├── deviceController.js
│       │   └── errorController.js
│       ├── middleware
│       │   ├── authMiddleware.js
│       │   └── validationMiddleware.js
│       ├── routes
│       │   ├── authRoutes.js
│       │   ├── deviceRoutes.js
│       │   └── errorRoutes.js
│       └── tests
│           ├── controllers
│           │   ├── authController.test.js
│           │   ├── deviceController.test.js
│           │   └── errorController.test.js
│           ├── middleware
│           │   ├── authMiddleware.test.js
│           │   └── validationMiddleware.test.mjs
│           └── routes
│               └── deviceRoutes.test.js
```



Technologies used

- Nodejs
  - Express
  - Express-validator
  - Bcryp
  - Mocha, chai, sinon, supertest
- Redis
- Postgresql
- Dependency track, Sonarcube, Trivy

Secure programming
credentials

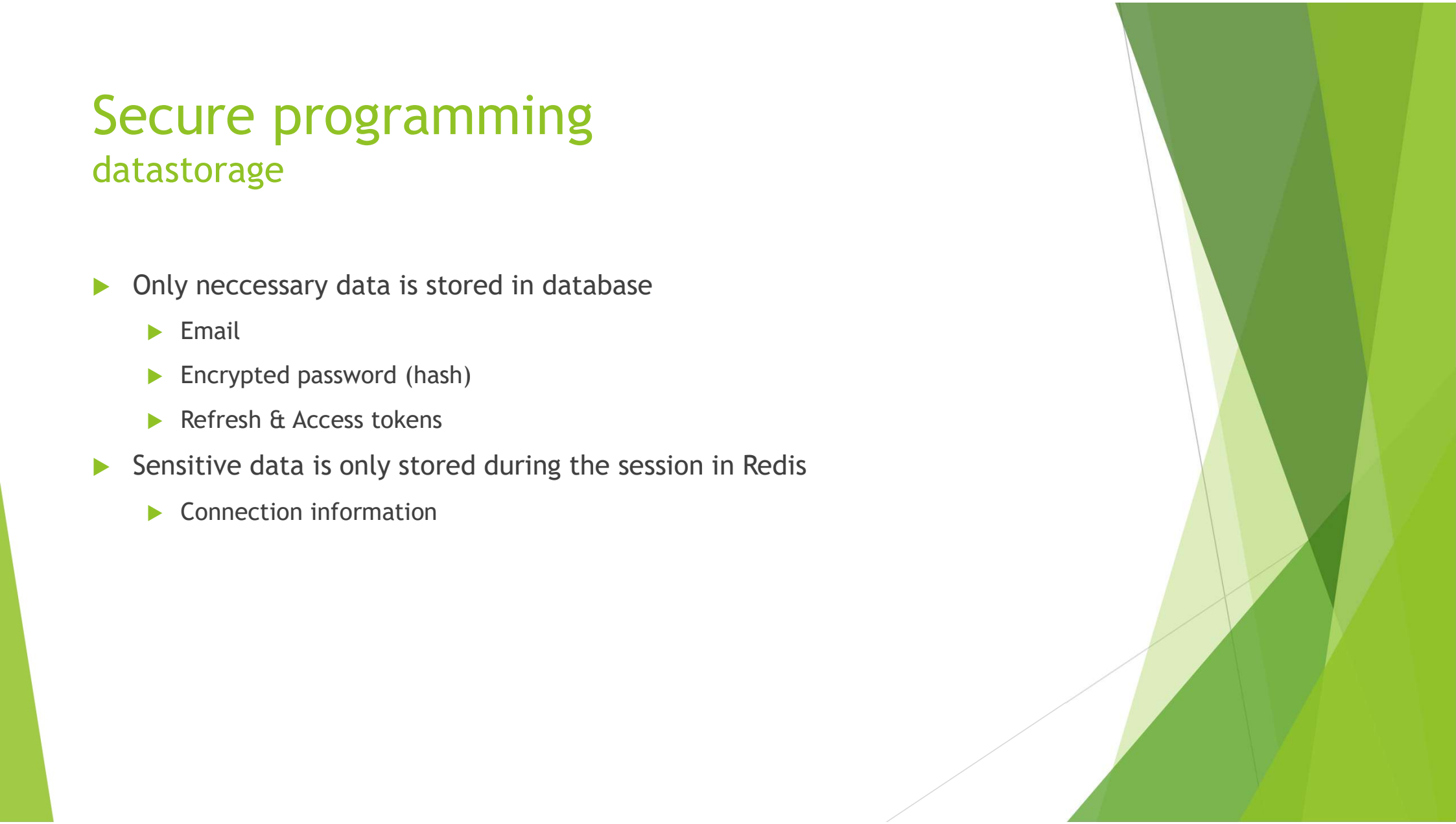
- Passwords are encrypted using bcrypt with 12 salt rounds
- JsonWebTokens are used for session handling

```
const generateTokens = (user) => {
  const payload = { id: user.id, email: user.email };
  const accessToken = jwt.sign(payload, process.env.ACCESS_TOKEN_SECRET, {
    expiresIn: "15m",
  });
  const refreshToken = jwt.sign(payload, process.env.REFRESH_TOKEN_SECRET, {
    expiresIn: "7d",
  });
  return { accessToken, refreshToken };
};
```

```
const storeRefreshToken = async (email, refreshToken) => {
  await pool.query(
    "INSERT INTO tokens (email, refresh_token) VALUES ($1, $2)",
    [email, refreshToken]
  );
};
```

Secure programming
datastorage

- Only neccessary data is stored in database
  - Email
  - Encrypted password (hash)
  - Refresh & Access tokens
- Sensitive data is only stored during the session in Redis
  - Connection information



Secure programming
protected endpoints

```
// Routes
app.use("/api/auth", authRoutes);
app.use("/api/device", deviceRoutes);
app.use("*", errorRoutes);
```

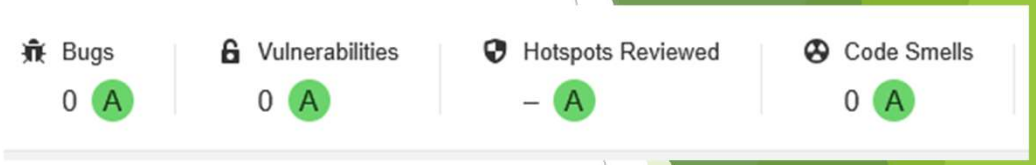
- Middlewares are used to protect the endpoints functions
- Only allowed endpoints are enabled (catch all)
- Ratelimit 15 connections / min

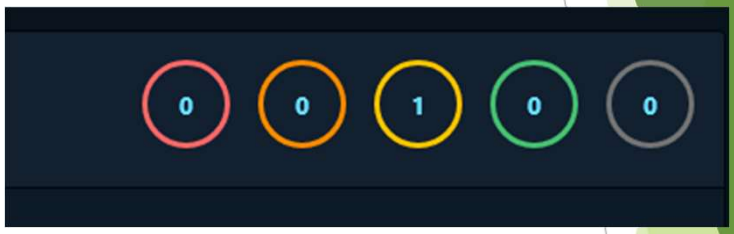
```
// Local Authentication
router.post(
  "/signup",
  whitelistFields(["email", "password"]),
  validateLoginCredentials,
  signup
);
router.post(
  "/login",
  whitelistFields(["email", "password"]),
  validateLoginCredentials,
  login
);
router.post(
  "/refresh",
  whitelistFields(["email", "refreshToken"]),
  refreshToken
);
router.post("/logout", deleteDevices, logout);

export default router;
```

```
export const validateEmail = body("email")
  .isEmail()
  .withMessage("Must be a valid email")
  .matches(/^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}$/)
  .withMessage("Email contains invalid characters")
  .isLength({ max: 254 })
  .withMessage("Email exceeds maximum length of 254 characters")
  .normalizeEmail();

export const whitelistFields = (allowedFields) => (req, res, next) => {
  const extraFields = Object.keys(req.body).filter(
    (f) => !allowedFields.includes(f)
  );
  if (extraFields.length > 0) {
    return res.status(400).json({
      error: `Unexpected fields: ${extraFields.join(", ")}`,
    });
  }
  next();
};
```

Testing

- Unit and integration testing was done manually
- Statical analysis with sonarcube
- Vulnerability scans with trivy and dependency track






```
449a08de36ac (alpine 3.21.3)
============================
Total: 0 (UNKNOWN: 0, LOW: 0, MEDIUM: 0, HIGH: 0, CRITICAL: 0)
```

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	77.92	84.37	75	77.92	
app	89.47	50	100	89.47	
server.js	89.47	50	100	89.47	33-36
app/src/config	88.46	50	100	88.46	
db.js	88.46	50	100	88.46	18,22-23
app/src/controllers	64.41	80.48	70.58	64.41	
authController.js	53.4	80	61.53	53.4	...-99,103-126,145-156,171-177
deviceController.js	91.42	79.16	100	91.42	52-54,67-69
errorController.js	100	100	100	100	
app/src/middleware	100	100	100	100	
authMiddleware.js	100	100	100	100	
validationMiddleware.js	100	100	100	100	
app/src/routes	100	100	100	100	
authRoutes.js	100	100	100	100	
deviceRoutes.js	100	100	100	100	
errorRoutes.js	100	100	100	100	

Thanks