# Unit Testing G1 Pet Adoption System – HappyTails

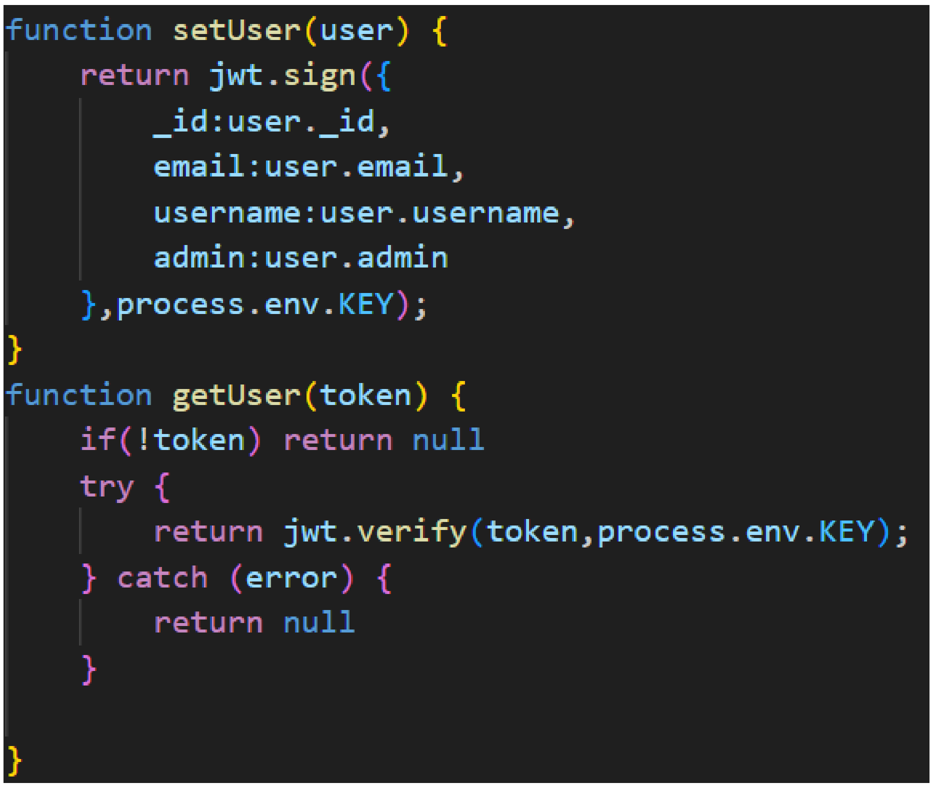
Testing framework used – Mocha @10.8.2

Assertion library used – Chai @4.3.4

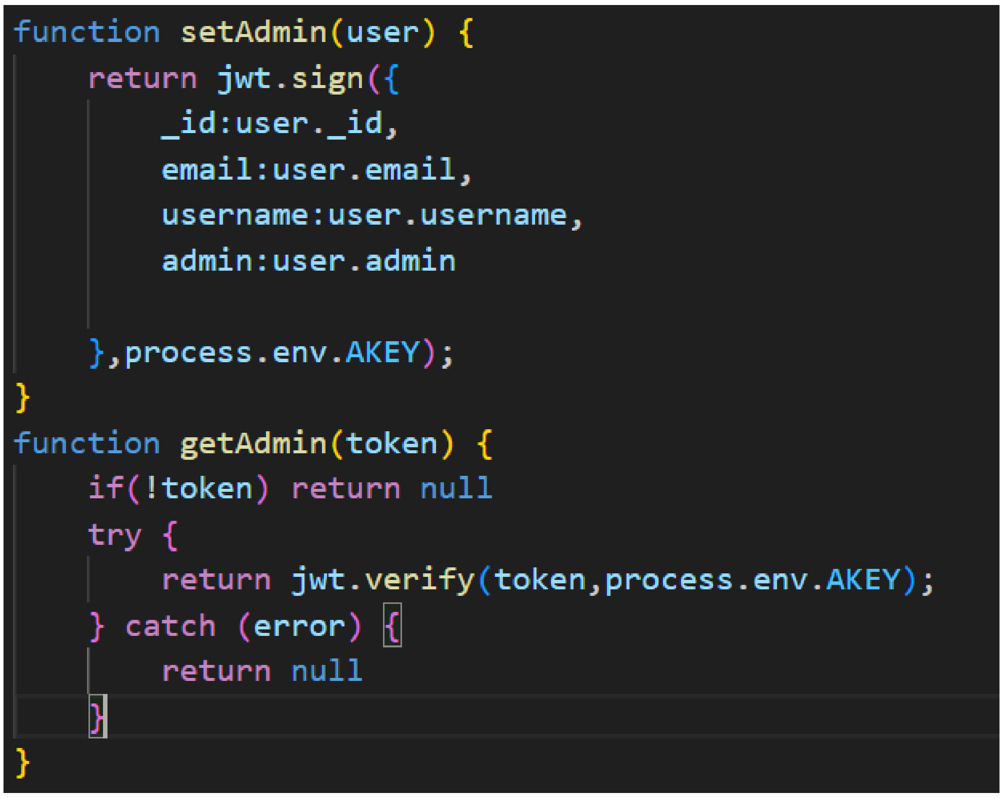
Other – Sinon @19.0.2

1. Auth.js

Source Code:

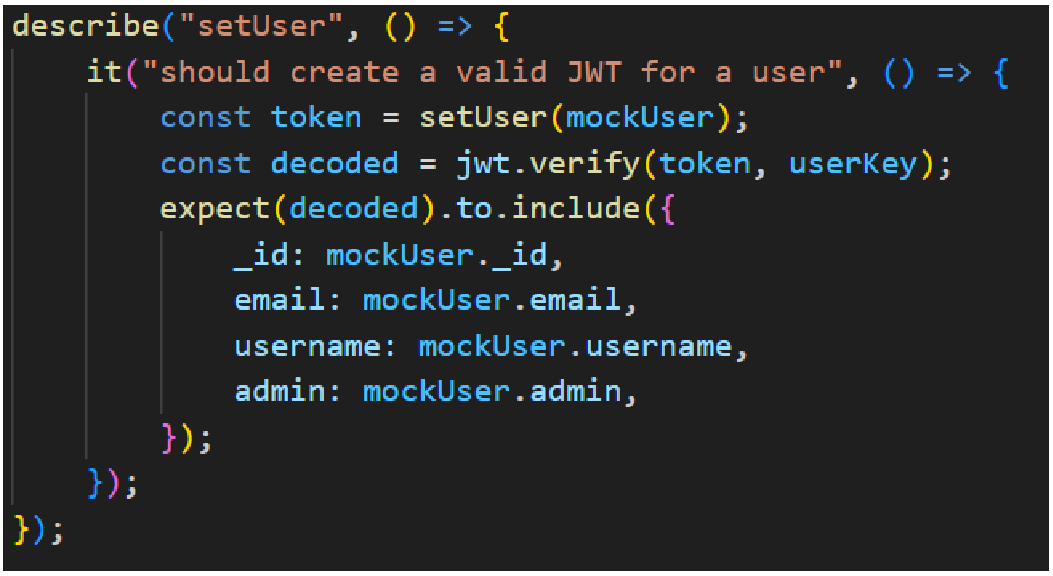


```
function setUser(user) {
    return jwt.sign({
        _id:user._id,
        email:user.email,
        username:user.username,
        admin:user.admin
    },process.env.KEY);
}
function getUser(token) {
    if(!token) return null
    try {
        return jwt.verify(token,process.env.KEY);
    } catch (error) {
        return null
    }

}
```



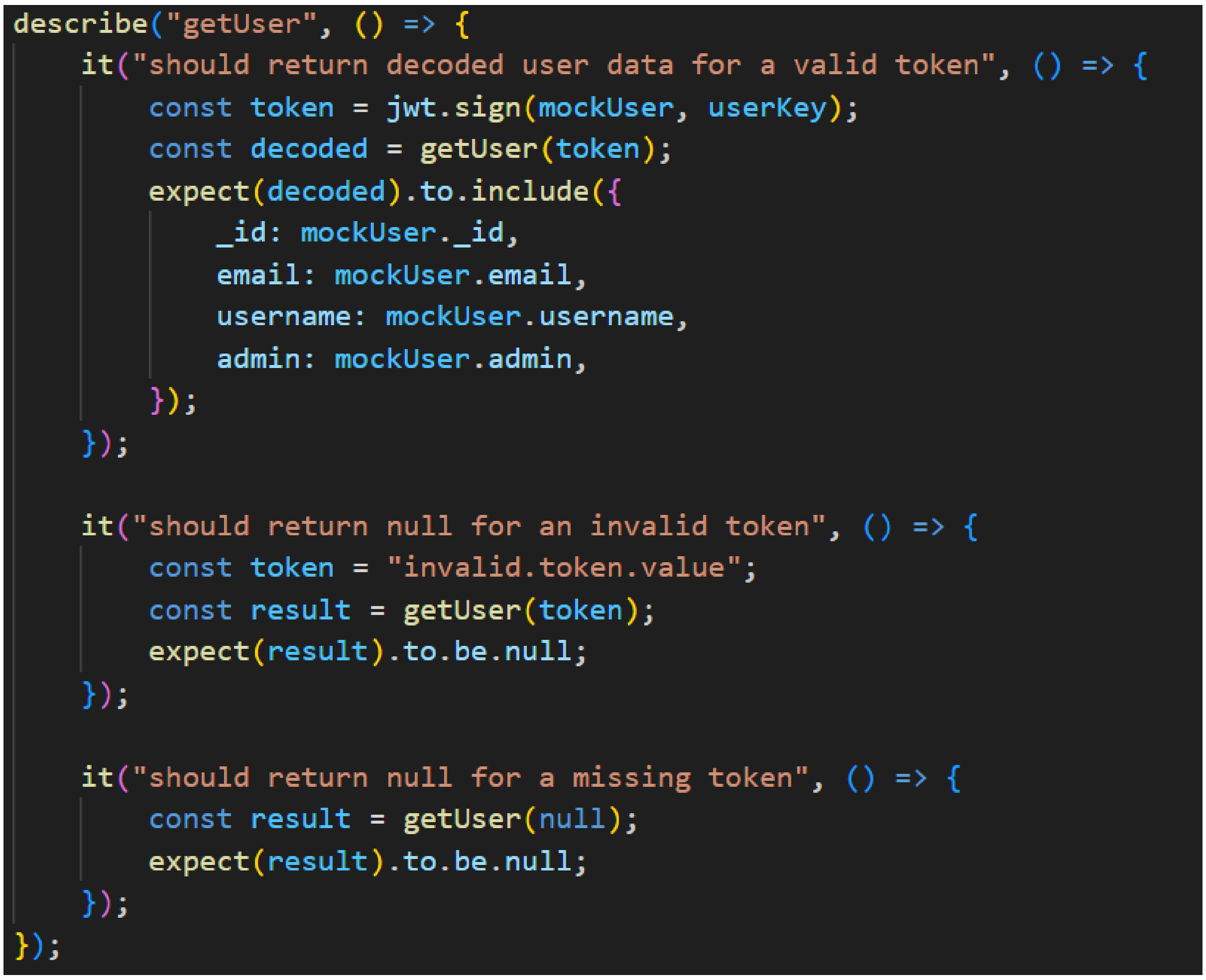
```
function setAdmin(user) {
    return jwt.sign({
        _id:user._id,
        email:user.email,
        username:user.username,
        admin:user.admin

    },process.env.AKEY);
}
function getAdmin(token) {
    if(!token) return null
    try {
        return jwt.verify(token,process.env.AKEY);
    } catch (error) {
        return null
    }
}
```

Test cases and results:



```
describe("setUser", () => {
    it("should create a valid JWT for a user", () => {
        const token = setUser(mockUser);
        const decoded = jwt.verify(token, userKey);
        expect(decoded).to.include({
            _id: mockUser._id,
            email: mockUser.email,
            username: mockUser.username,
            admin: mockUser.admin,
        });
    });
});
```

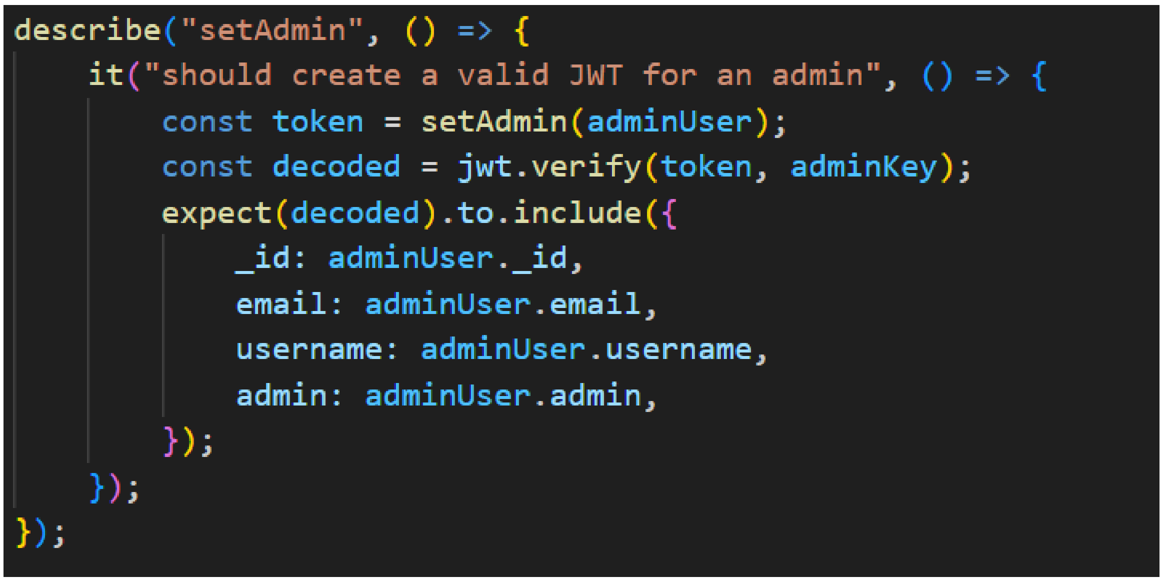
This case checks the condition where the user is JWT verified and a token is created.



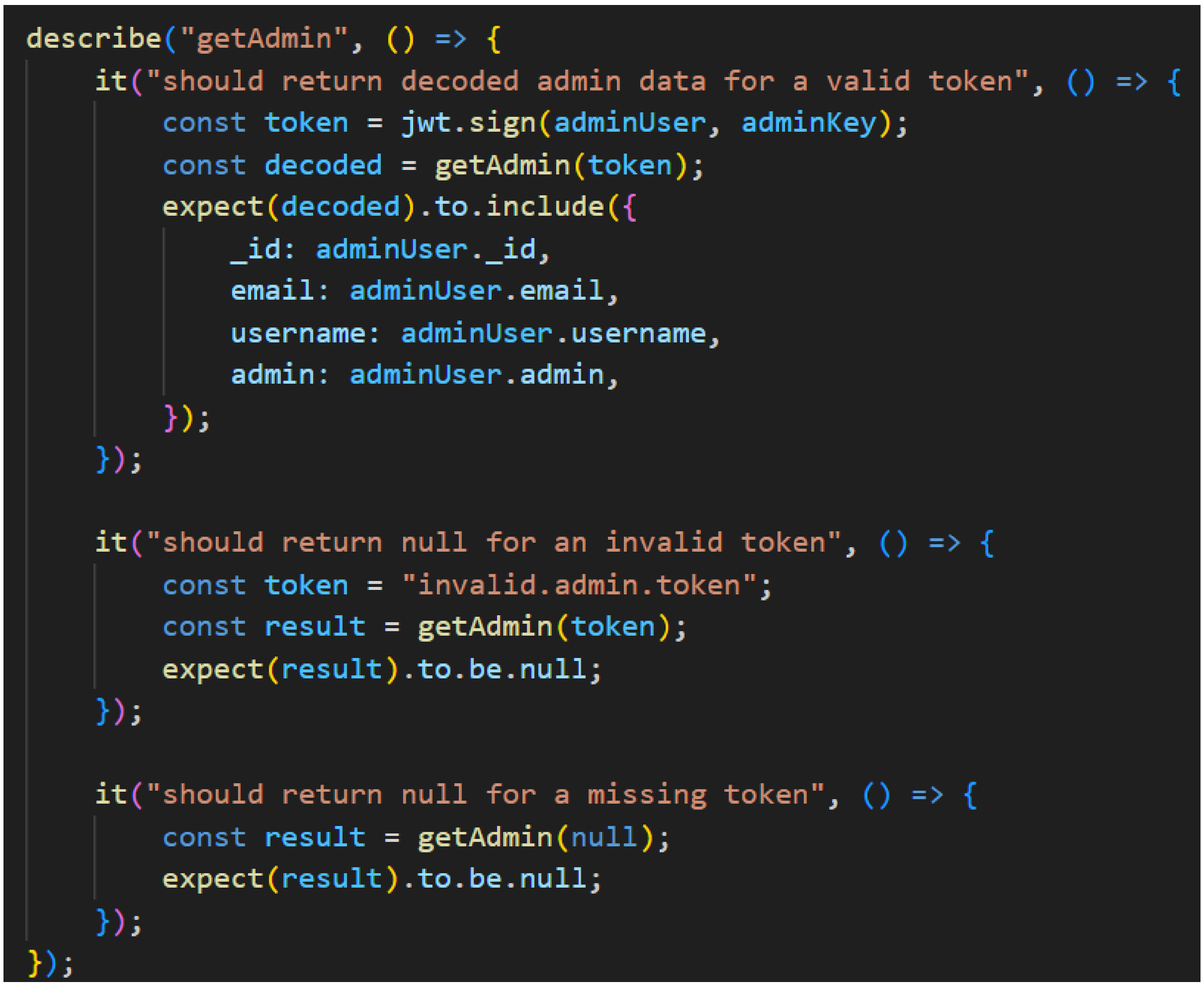
```
describe("getUser", () => {
    it("should return decoded user data for a valid token", () => {
        const token = jwt.sign(mockUser, userKey);
        const decoded = getUser(token);
        expect(decoded).to.include({
            _id: mockUser._id,
            email: mockUser.email,
            username: mockUser.username,
            admin: mockUser.admin,
        });
    });

    it("should return null for an invalid token", () => {
        const token = "invalid.token.value";
        const result = getUser(token);
        expect(result).to.be.null;
    });

    it("should return null for a missing token", () => {
        const result = getUser(null);
        expect(result).to.be.null;
    });
});
```

These test cases check both the conditions where the user is returned after the token is verified, where an invalid token is received and where the token is missing respectively

Similar test cases are made for the following function which are quite like these test cases.

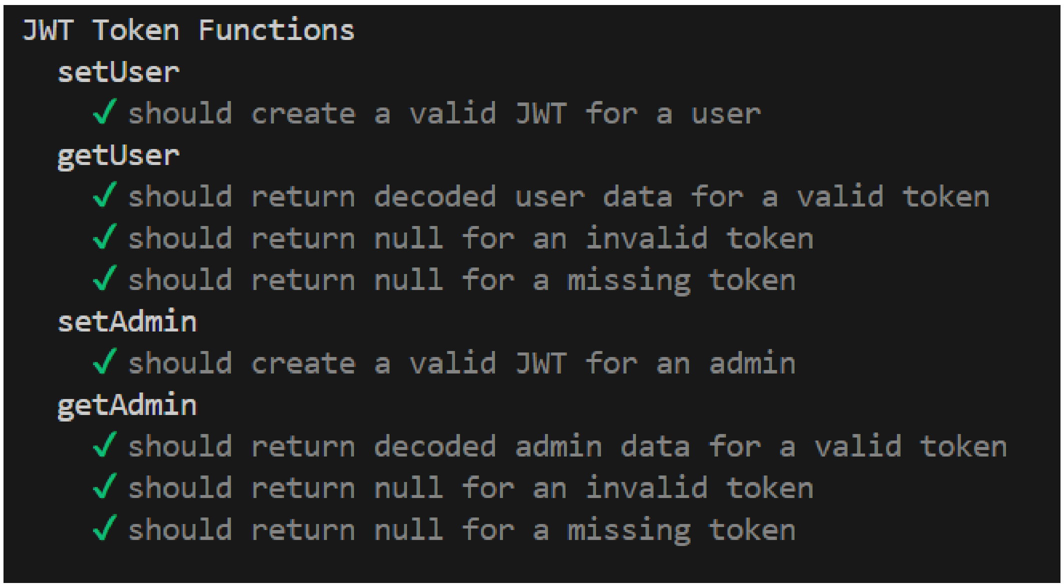


```
describe("setAdmin", () => {
    it("should create a valid JWT for an admin", () => {
        const token = setAdmin(adminUser);
        const decoded = jwt.verify(token, adminKey);
        expect(decoded).to.include({
            _id: adminUser._id,
            email: adminUser.email,
            username: adminUser.username,
            admin: adminUser.admin,
        });
    });
});
```



```javascript
describe("getAdmin", () => {
    it("should return decoded admin data for a valid token", () => {
        const token = jwt.sign(adminUser, adminKey);
        const decoded = getAdmin(token);
        expect(decoded).to.include({
            _id: adminUser._id,
            email: adminUser.email,
            username: adminUser.username,
            admin: adminUser.admin,
        });
    });

    it("should return null for an invalid token", () => {
        const token = "invalid.admin.token";
        const result = getAdmin(token);
        expect(result).to.be.null;
    });

    it("should return null for a missing token", () => {
        const result = getAdmin(null);
        expect(result).to.be.null;
    });
});
```

Results:

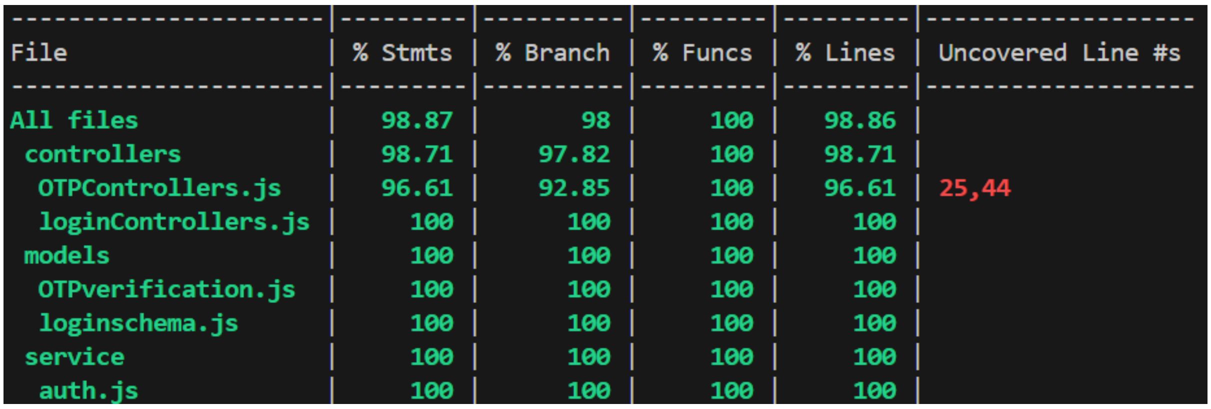


```
JWT Token Functions
  setUser
    ✓ should create a valid JWT for a user
  getUser
    ✓ should return decoded user data for a valid token
    ✓ should return null for an invalid token
    ✓ should return null for a missing token
  setAdmin
    ✓ should create a valid JWT for an admin
  getAdmin
    ✓ should return decoded admin data for a valid token
    ✓ should return null for an invalid token
    ✓ should return null for a missing token
```

The test cases are passed and give the outcomes as expected.

Coverage:

The test coverage for this test suite is 100 percent and statements, branches and functions are all covered 100 percent.



```
----------------------|---------|----------|---------|---------|-------------------
File                  | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
----------------------|---------|----------|---------|---------|-------------------
All files             |   98.87 |       98 |     100 |   98.86 |
 controllers          |   98.71 |    97.82 |     100 |   98.71 |
  OTPControllers.js   |   96.61 |    92.85 |     100 |   96.61 | 25,44
  loginControllers.js |     100 |      100 |     100 |     100 |
 models               |     100 |      100 |     100 |     100 |
  OTPverification.js  |     100 |      100 |     100 |     100 |
  loginschema.js      |     100 |      100 |     100 |     100 |
 service              |     100 |      100 |     100 |     100 |
  auth.js             |     100 |      100 |     100 |     100 |
```