Project Meeting 2 Requirements:

1. Is there any change of refinement in the requirement since the last project meeting?

- Identified and clearly discussed in the project report
- If no change, mention

Clarification of Requirement with the Accenture:

1) Admins and users are defined as the ACNAPI team and customers or potential customers of ACNAPI respectively.
2) Building to support the ACNAPI platform and their current only way is the Contact Us form.

2. Detailed description of use-case diagram

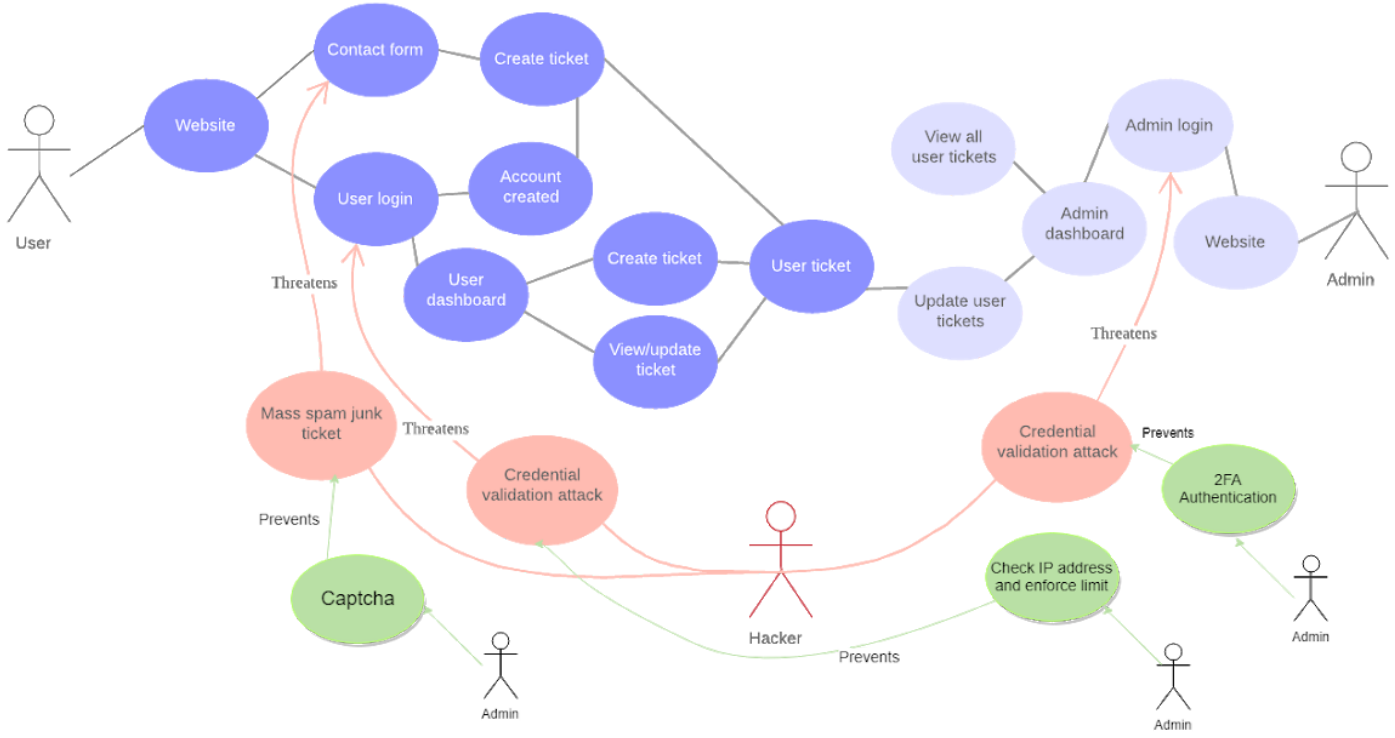


Figure 1: Use case diagram of the main features of the product

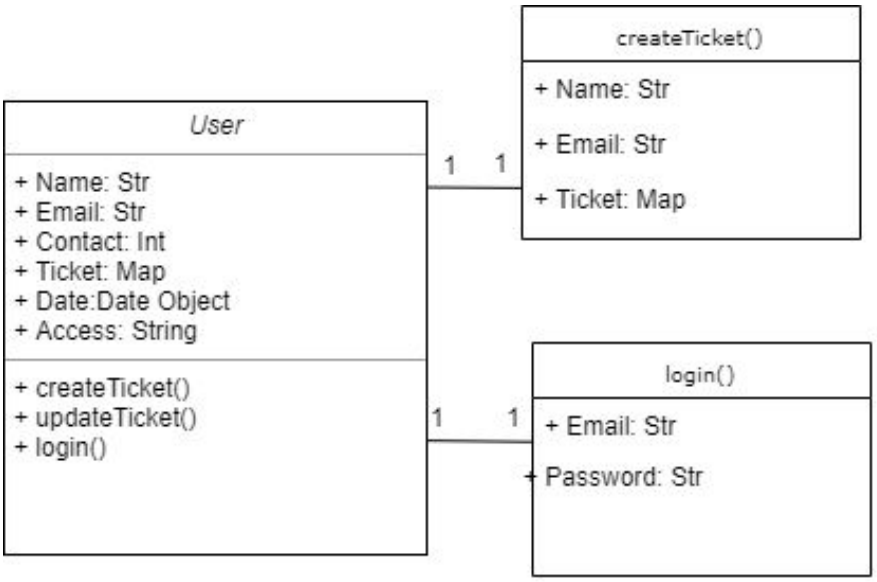


Figure 2: Class diagram for our login feature

How is the product going to be tested

- No coding is required but a clear plan for testing is highly desired
    - Tools for unit testing and robustness
    - Examples are included for each type of testing
    - Robustness testing should correlate to misuse case diagrams and other tests should correspond to use cases in use case diagram

a) Testing the Frontend (React.js)

- Testing for react.js we will be using jest and enzyme

1. `npm install --save-dev enzyme enzyme-adapter-react-16`
2. Need to create a test folder with <filename>.test.js

Testing existence of button:

```
describe('Login', ()=>{
    it('button found', ()=>{
        const wrapper = shallow(<Login />);
        const button = wrapper.find('button');
        button.simulate('click');
    });
})
```

Physical simulation testing tools: **Selenium**, for open source software to automate browser clicks to go to the website then automatically click register/login

b) Testing the Backend (Node.js)

- Testing for node.js we will be using mocha and chai, very similar to java syntax for testing. Some key functions to call from these libs for testing is describe(), it(), assert()

1. `npm install mocha chai --save-dev`
2. Create a folder call ‘test’, mocha will be looking for a folder called test
3. The files created in the test folder should be <filename_that_create_test_for>Test.js

Testing for results of that module:

```
describe('App', function(){
    it('App should return hello', function(){
        assert.equal(app(),'hello');
    });
})
```

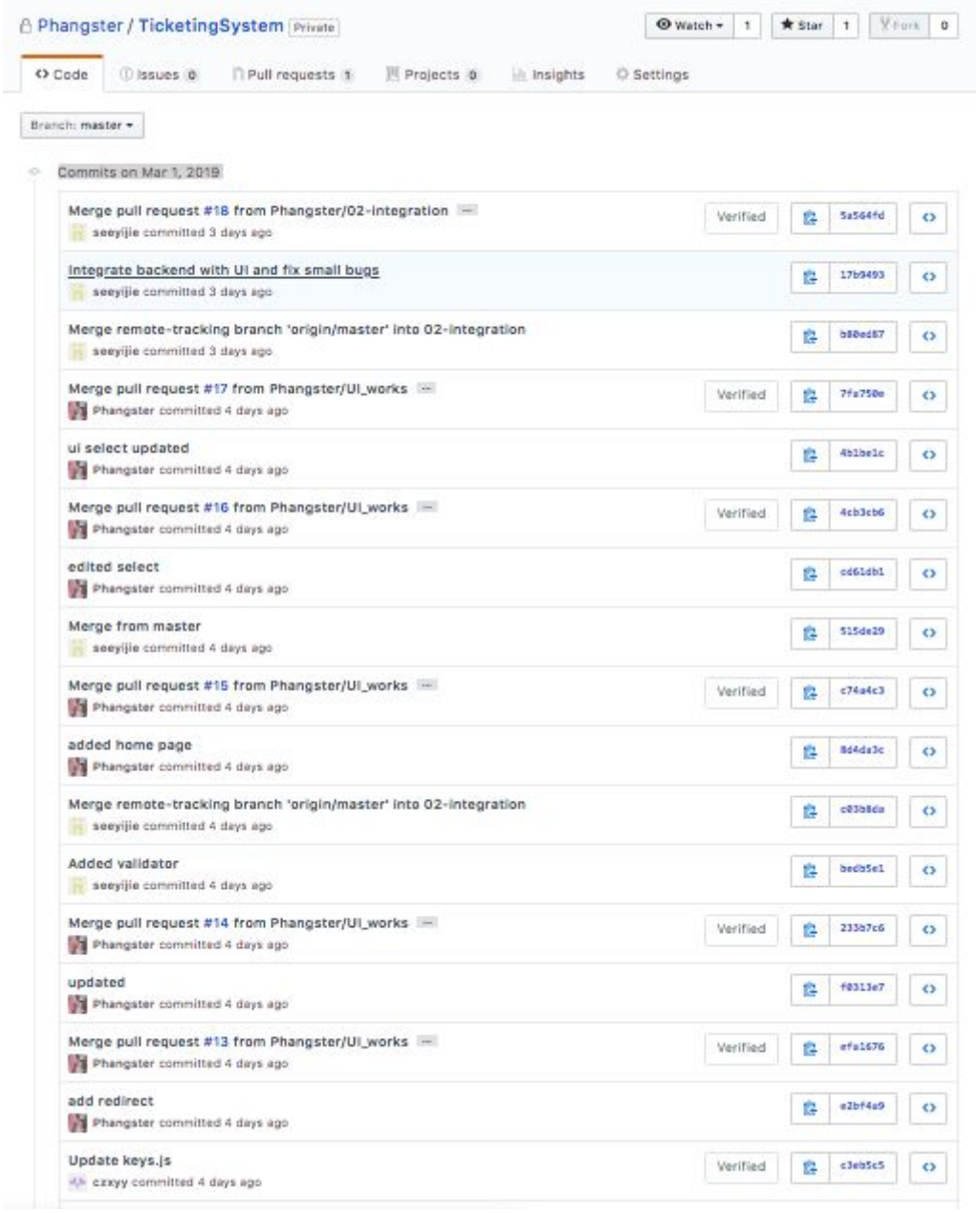
API testing for response code:

```
import chai from 'chai';
import chaiHttp from 'chai-http';
import server from '../server';

chai.use(chaiHttp);
let should = chai.should();

describe('Testing for http', function(done){
    it('Successful', function(){
        chai.request(server)
            .get('/login')
            .end(function(err,res){
                res.should.have.status(200)
                done()
            })
    });
})
```

The case of success of registering or login: response code will indicate 200

The case of failure: response code will indicate 400/404 (4XX)

6. Evidence of commit record



https://github.com/Phangster/TicketingSystem/commits/master