ASSIGNMENT 2

BIT 302

Software Engineering

Design Document and Tests Document

"Web-based University Registration Management Information System"

Prepared by:

Team Leader: I Putu Wahyu Adhi Diatmika (E1600725)

Contact Details: wahyuadhi03@gmail.com

Team Member: Ida Ayu Ritra Prabandari (E1600727)

Contact Details: idaayuritra21@gmail.com

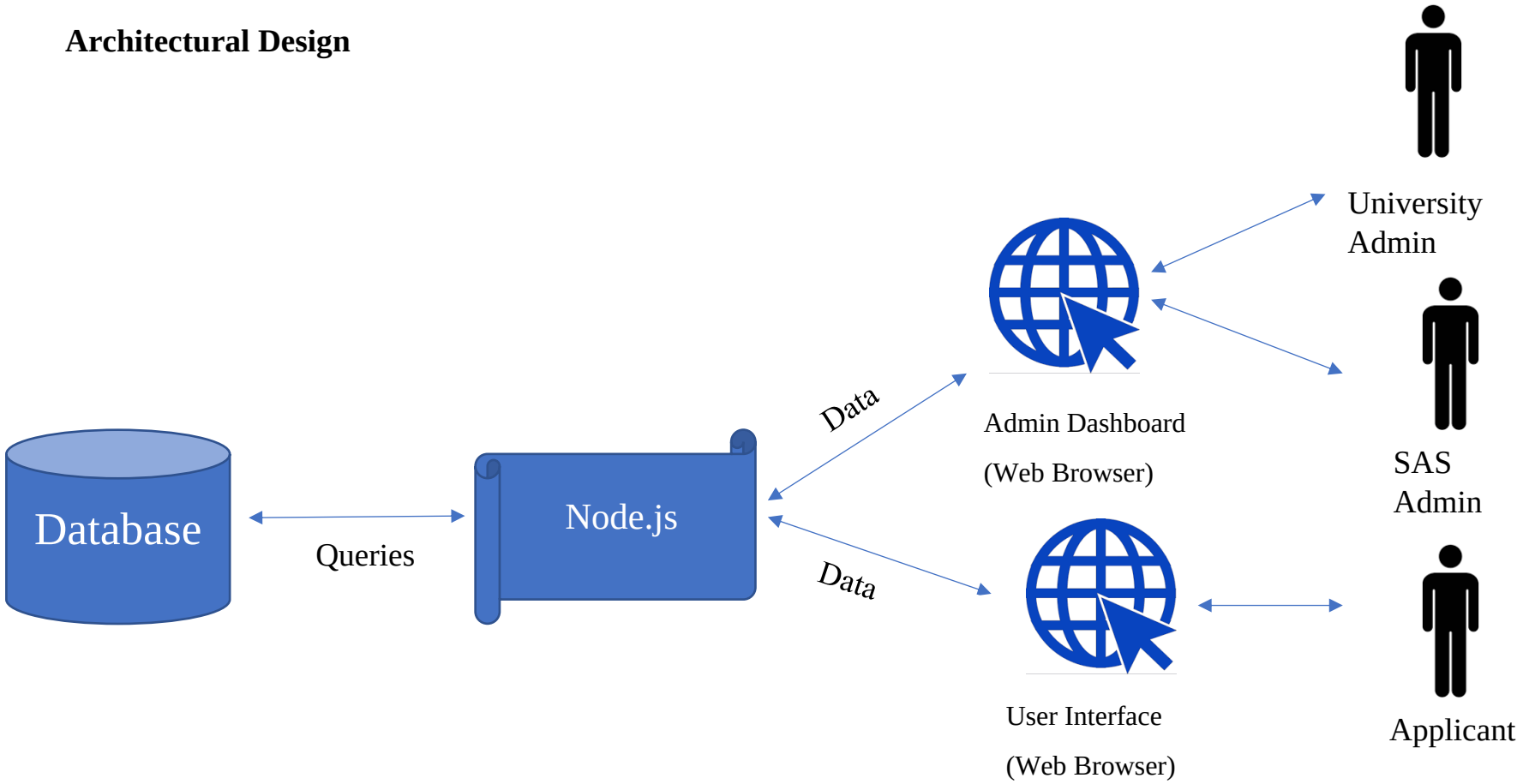
Table of Contents

Introduction

Every application that wants to be developed because of problems faced by users or clients that can be solved through information technology. It is important to know the requirements needed to solve the problem and the way the system works to help users or clients. Requirements are known so solutions are found to solve the problem. The solution is called design. Therefore, a design is needed in developing an application.

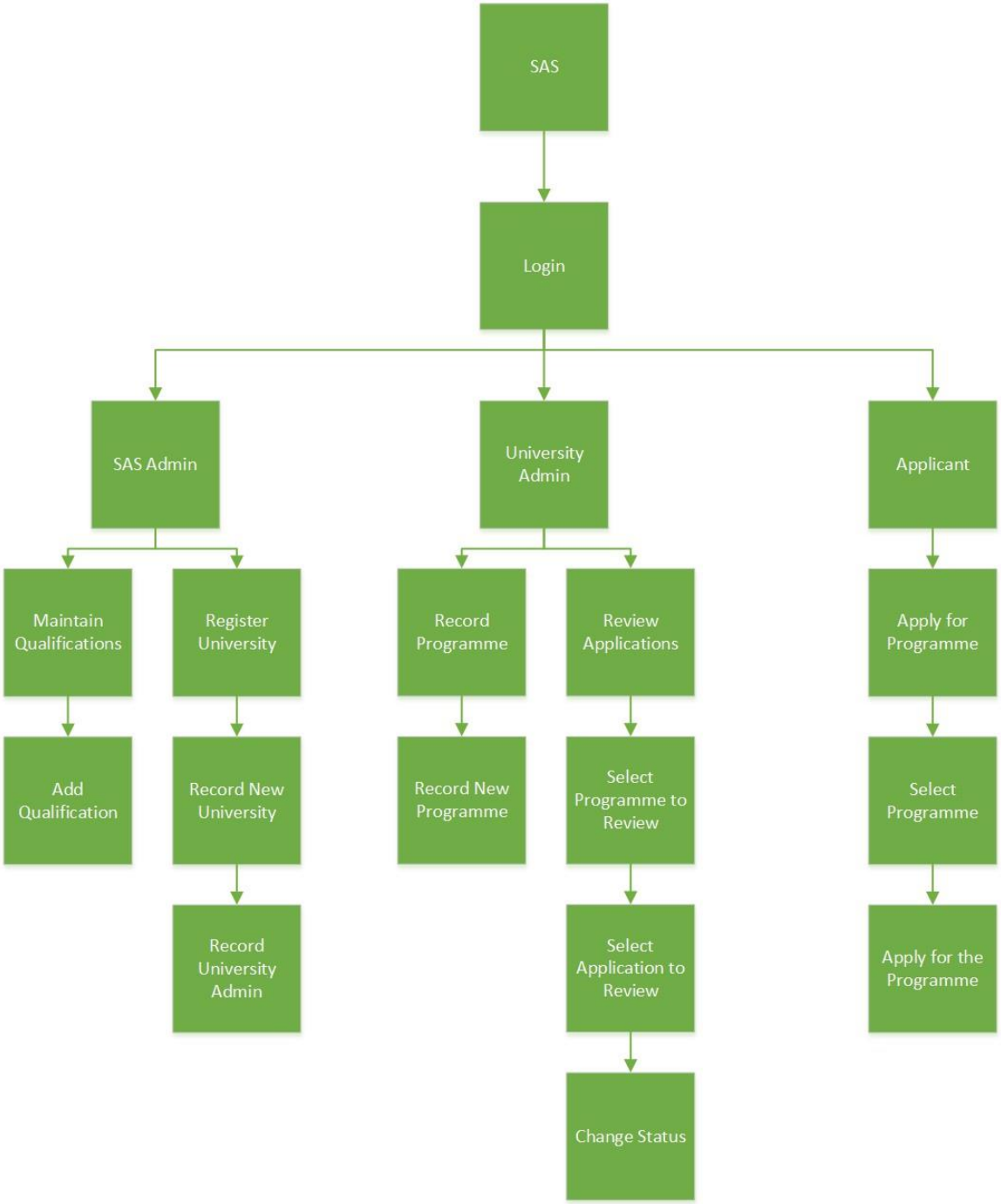
In this task will be related to design, we will produce Architectural design, Design class diagrams, System sequence class diagrams, Data design (ERD), Database design, and Wire frame.

Architectural Design



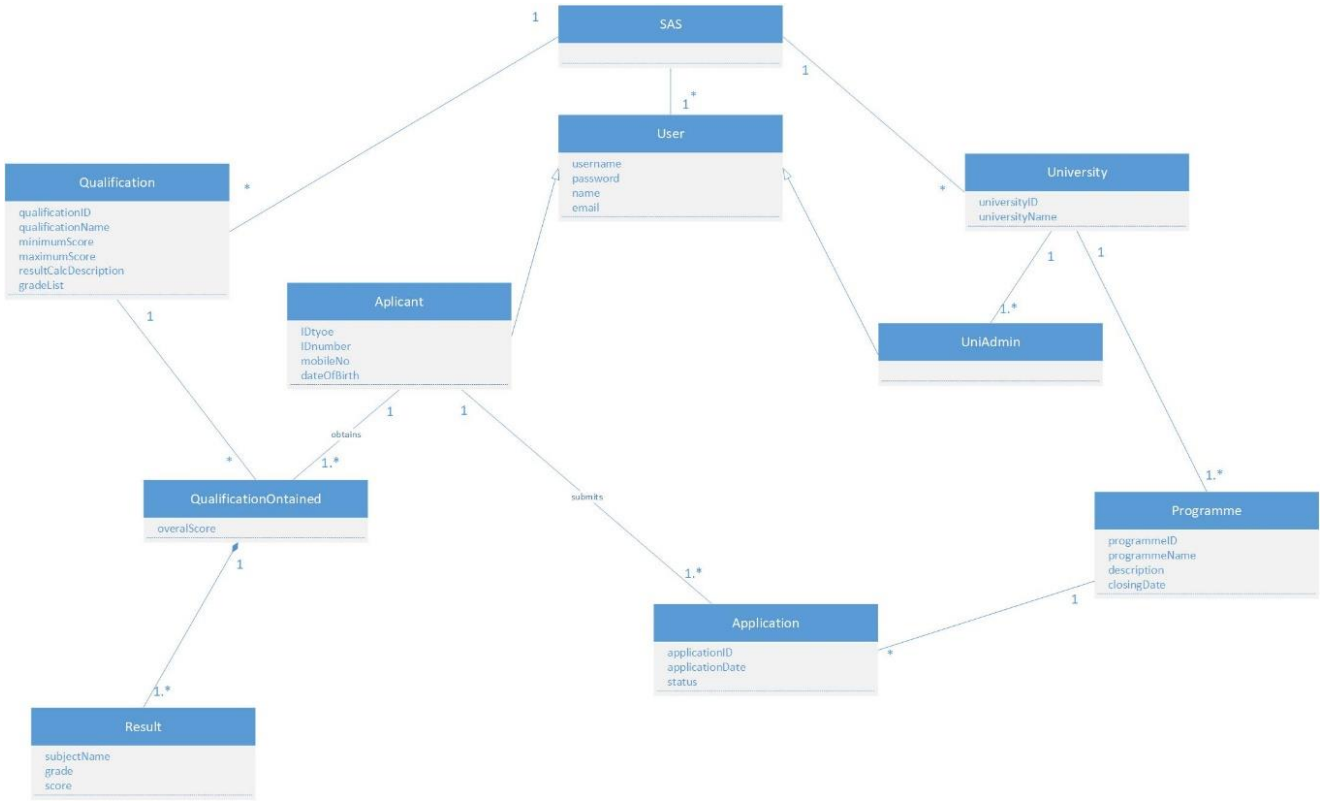
Picture 1: Architectural Design

Sitemap



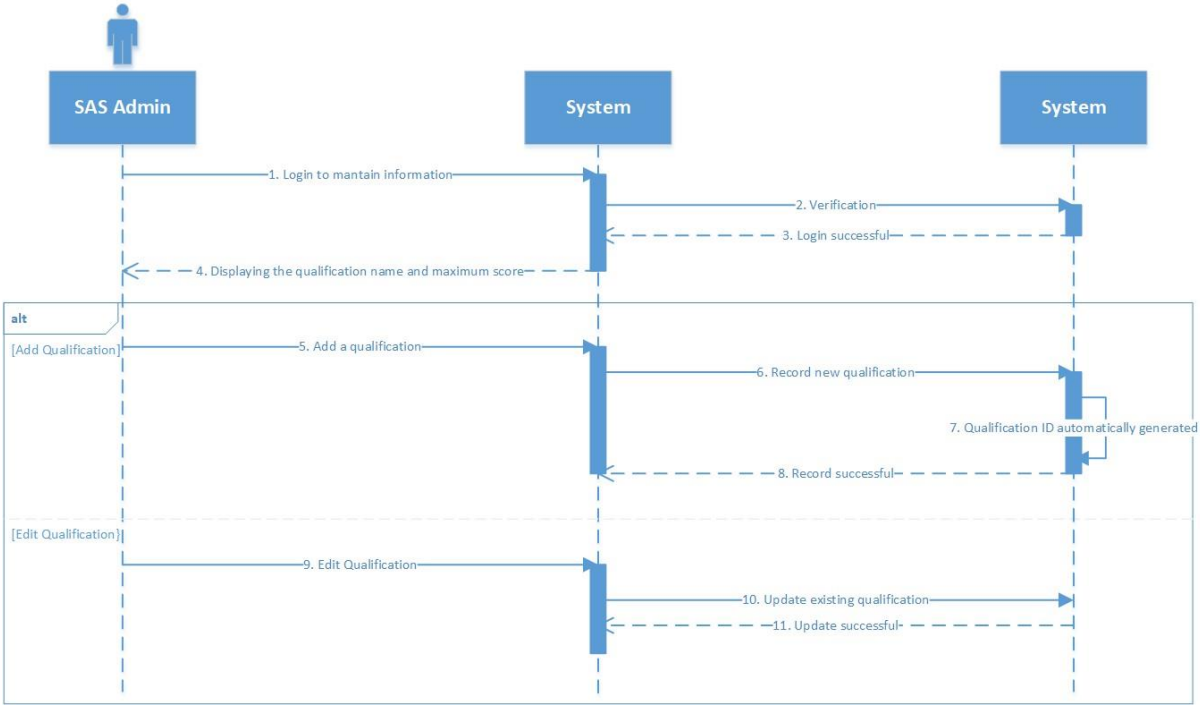
Picture 2: Site Map

Design Class Diagram

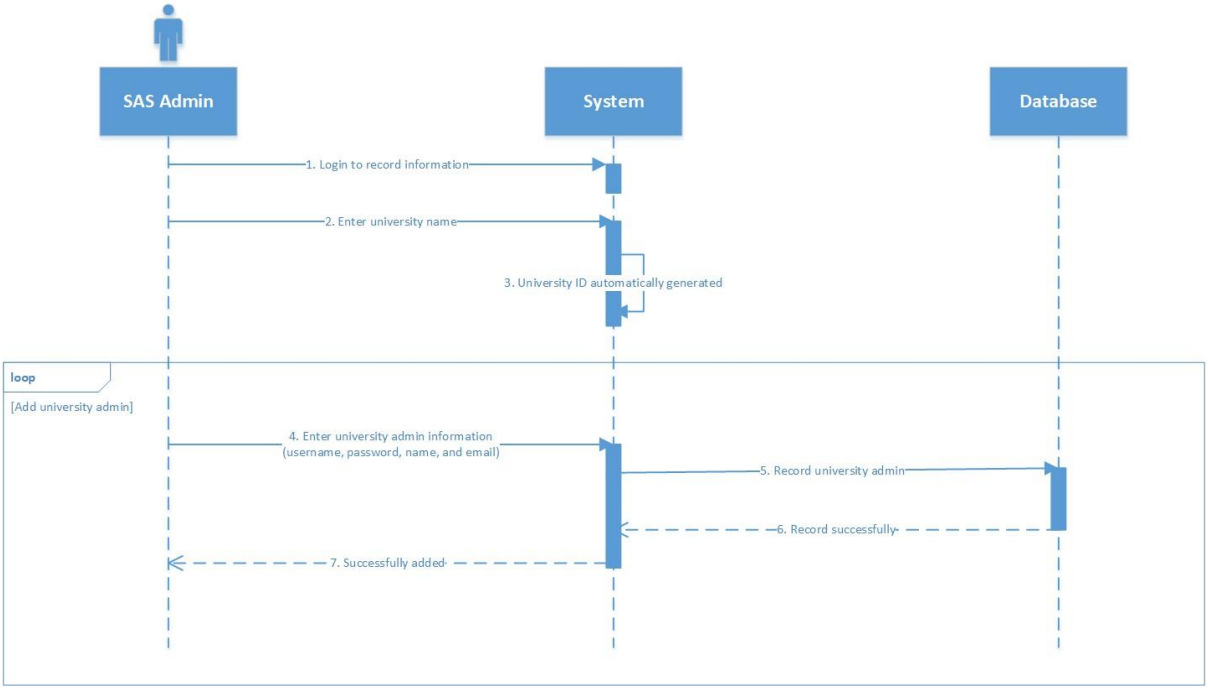


Picture 3: Class Diagram

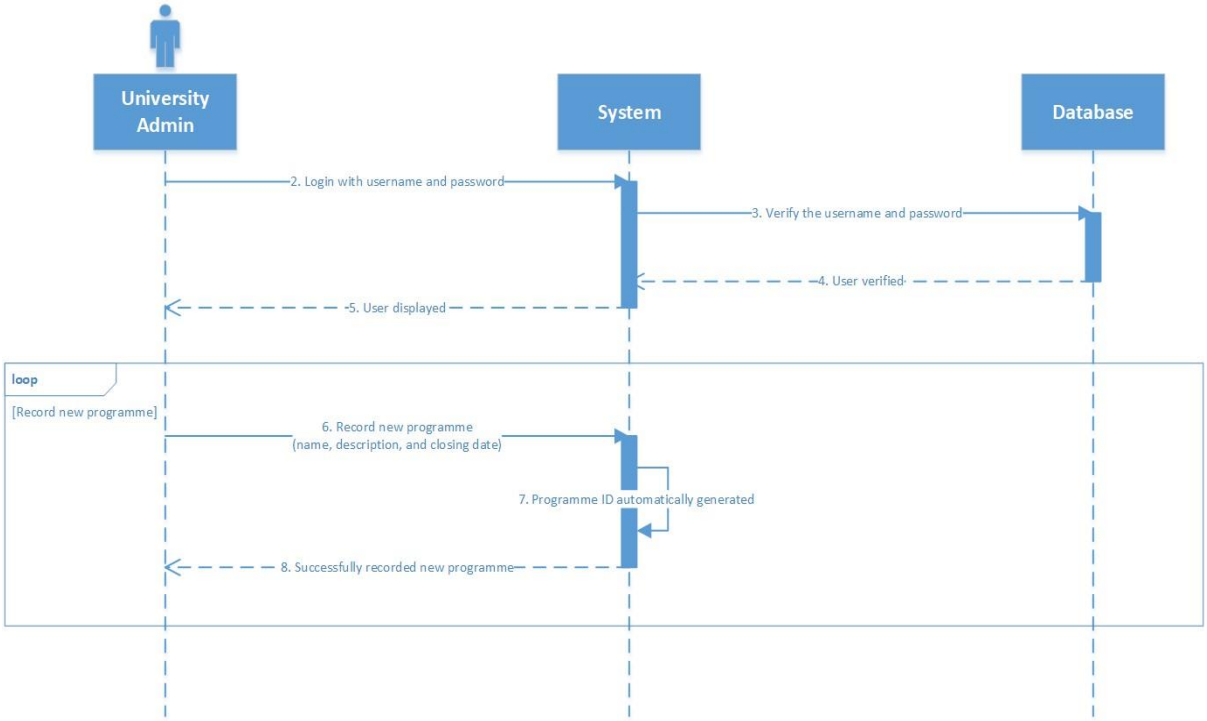
System Sequence Diagram



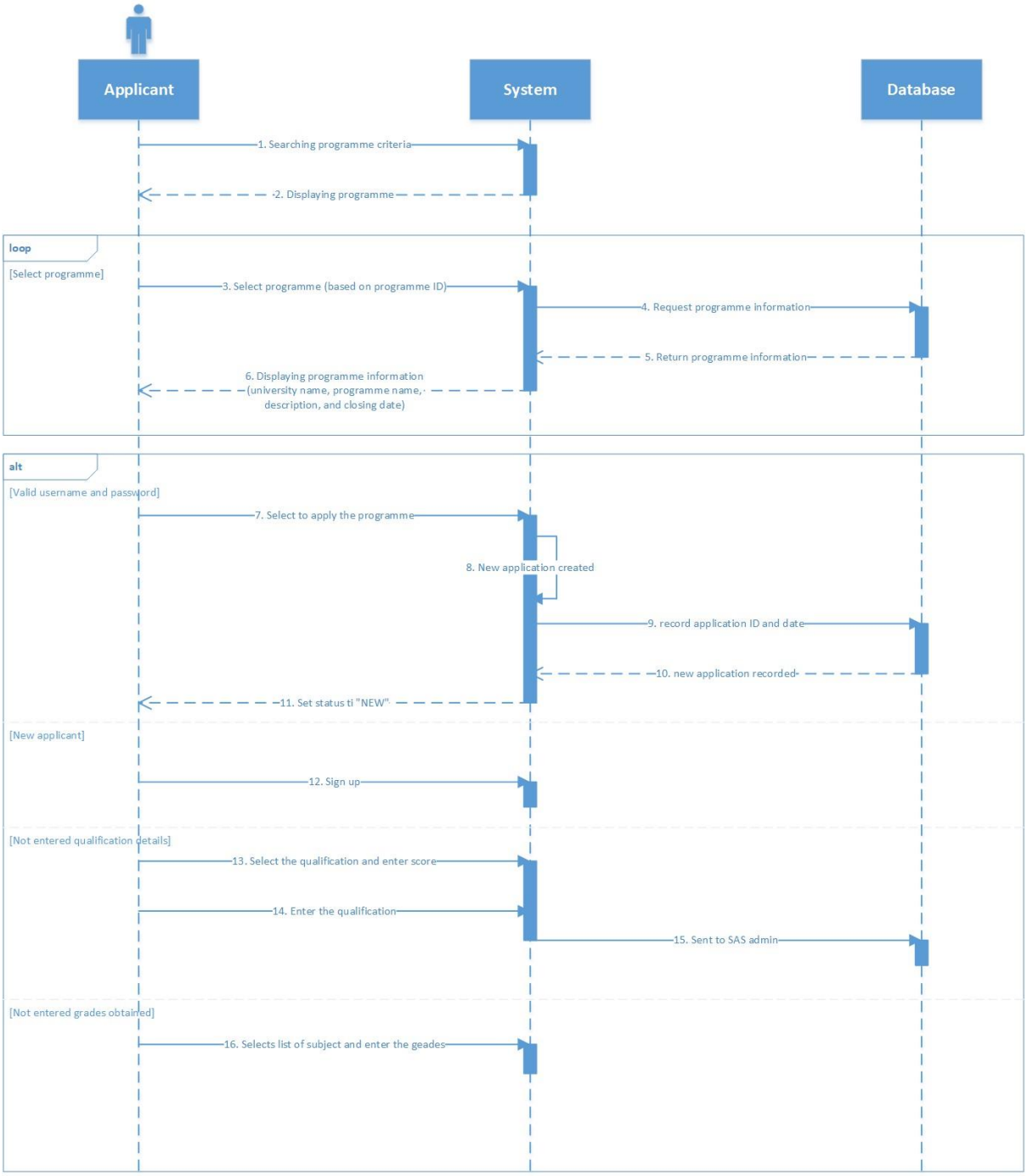
Picture 4: System Sequence Diagram of Maintain Qualifications



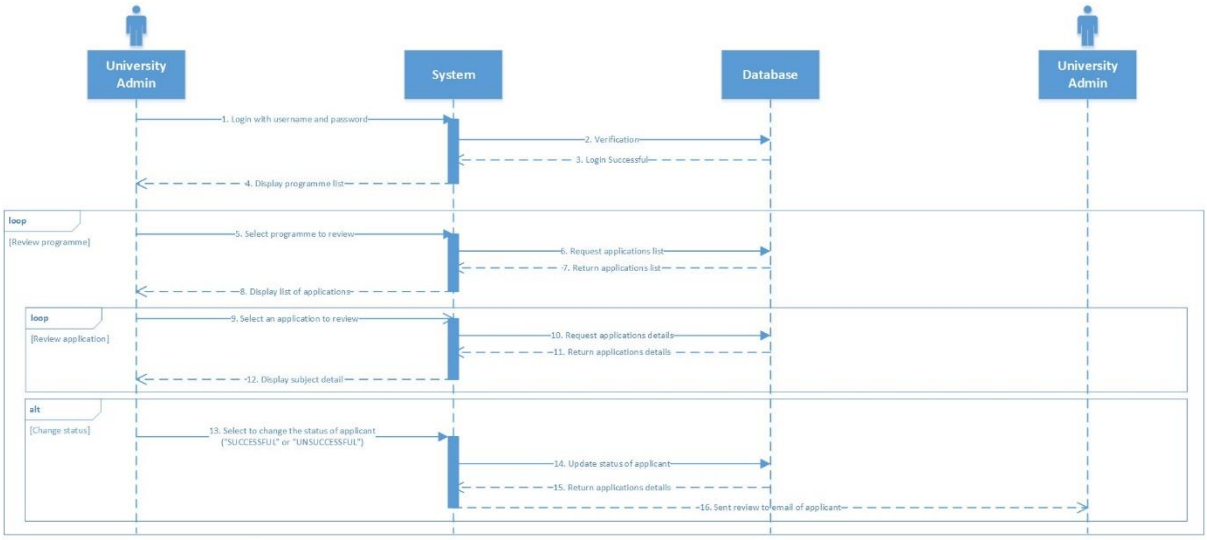
Picture 5: System Sequence Diagram of Register University



Picture 6: System Sequence Diagram of Record Programme

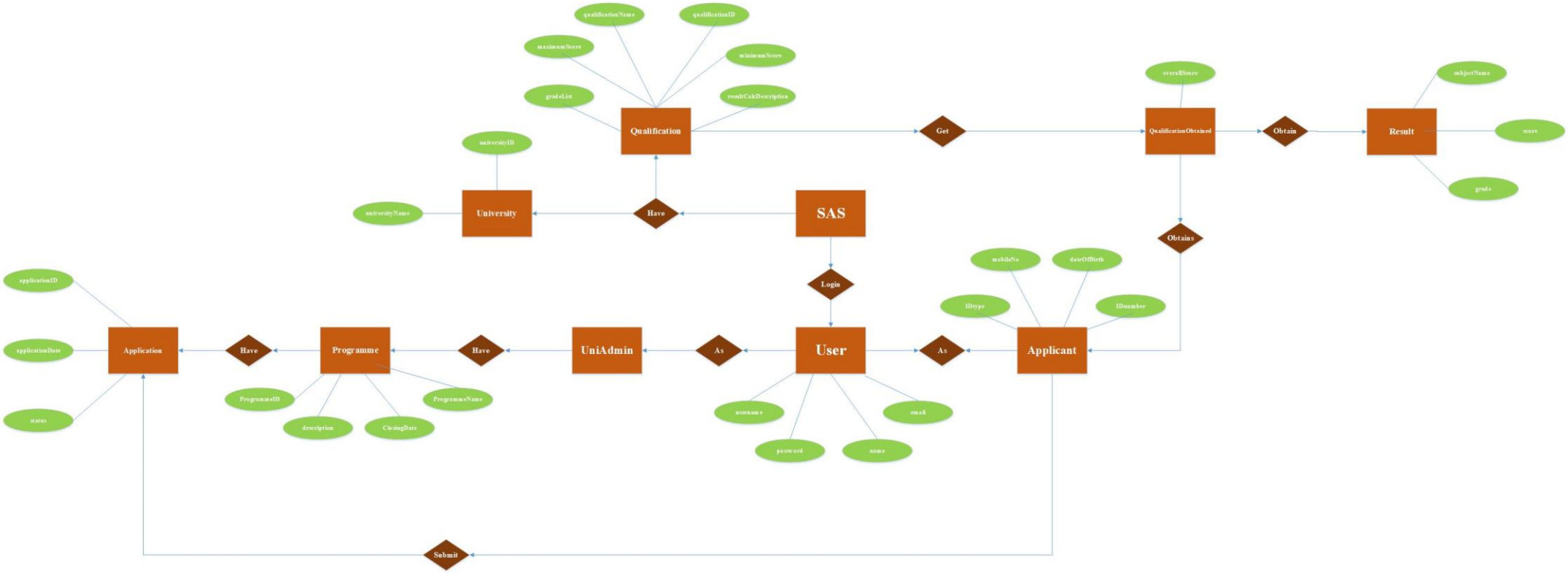


Picture 7: System Sequence Diagram of Apply for Programme



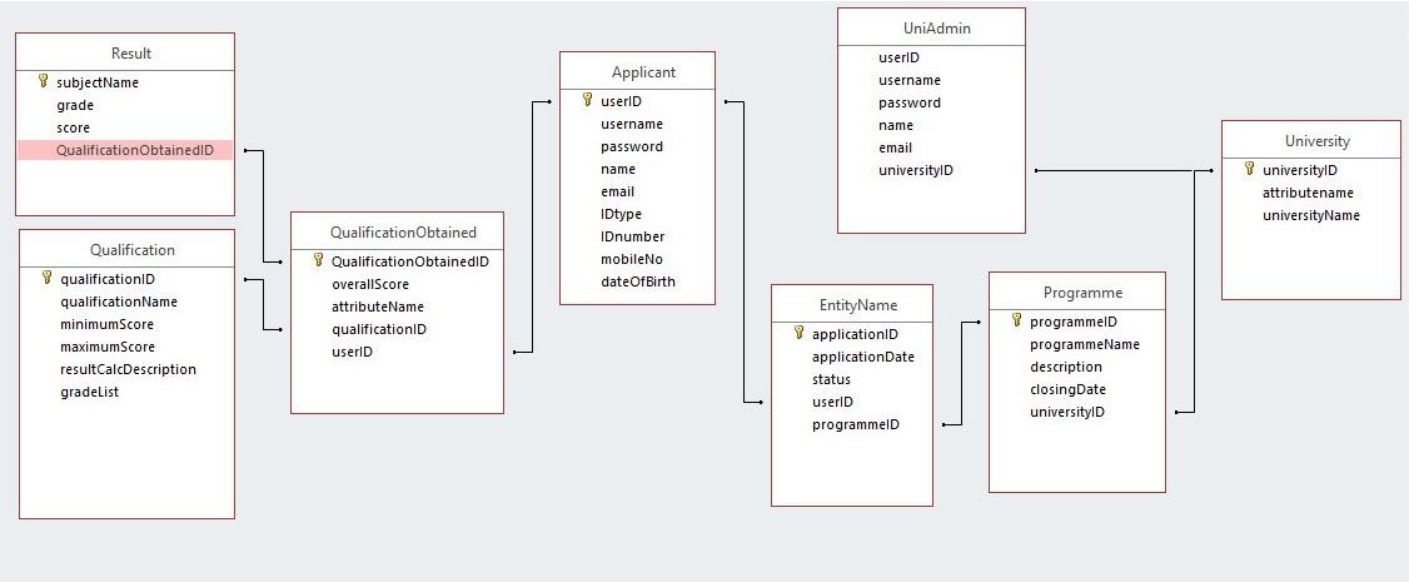
Picture 8: System Sequence Diagram of Review Applications

Data Design (ERD)



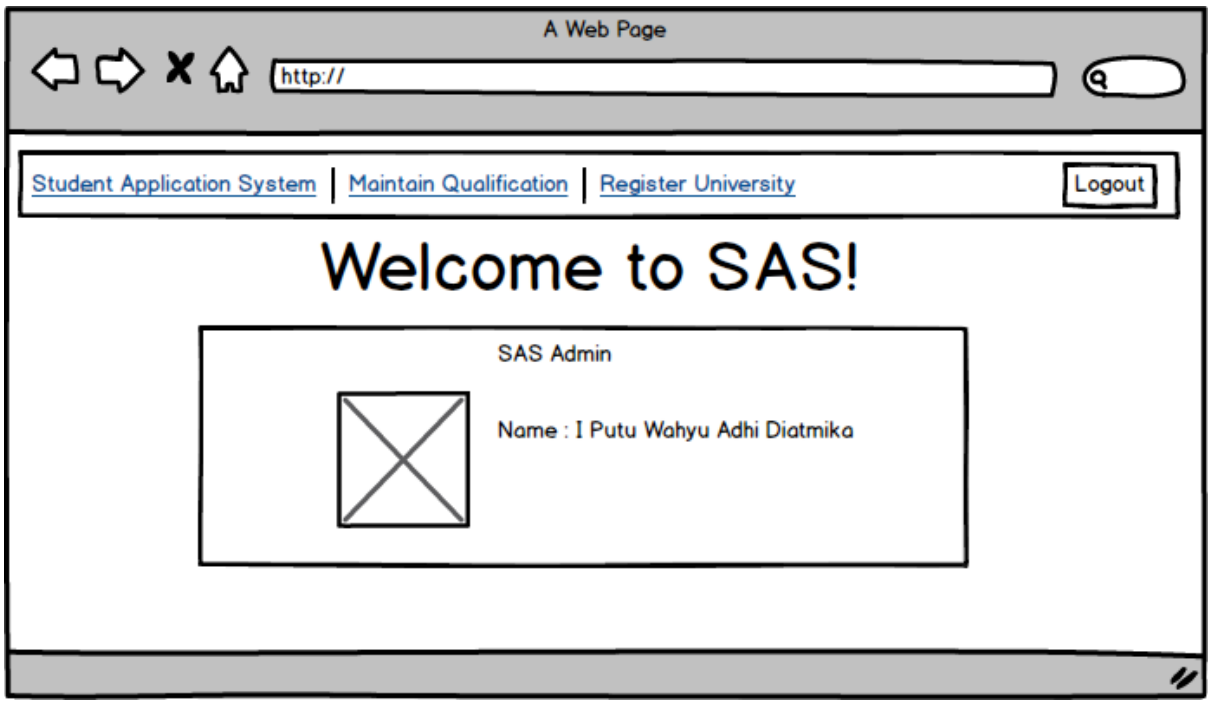
Picture 9: Data Design (ERD)

Database Design

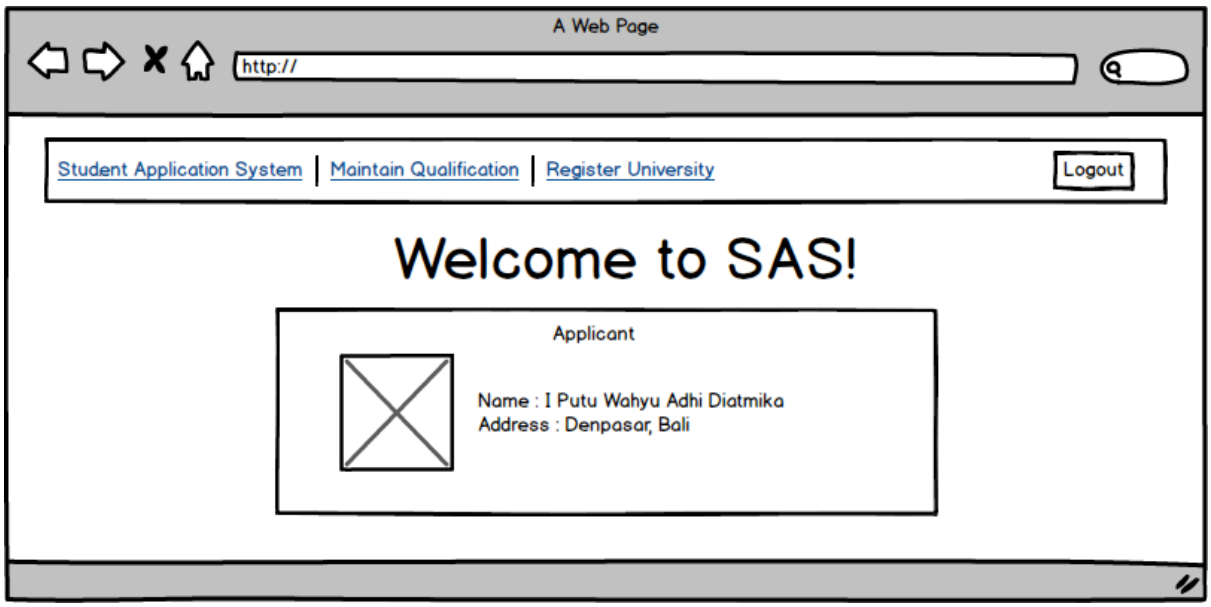


Picture 10: Database Design

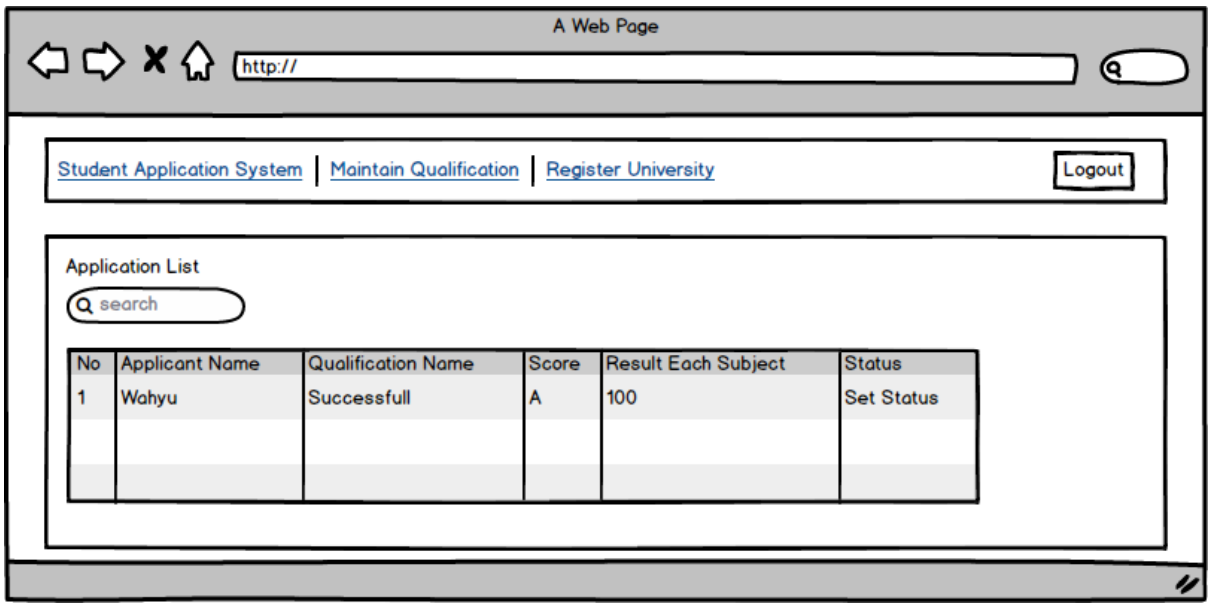
Wireframe



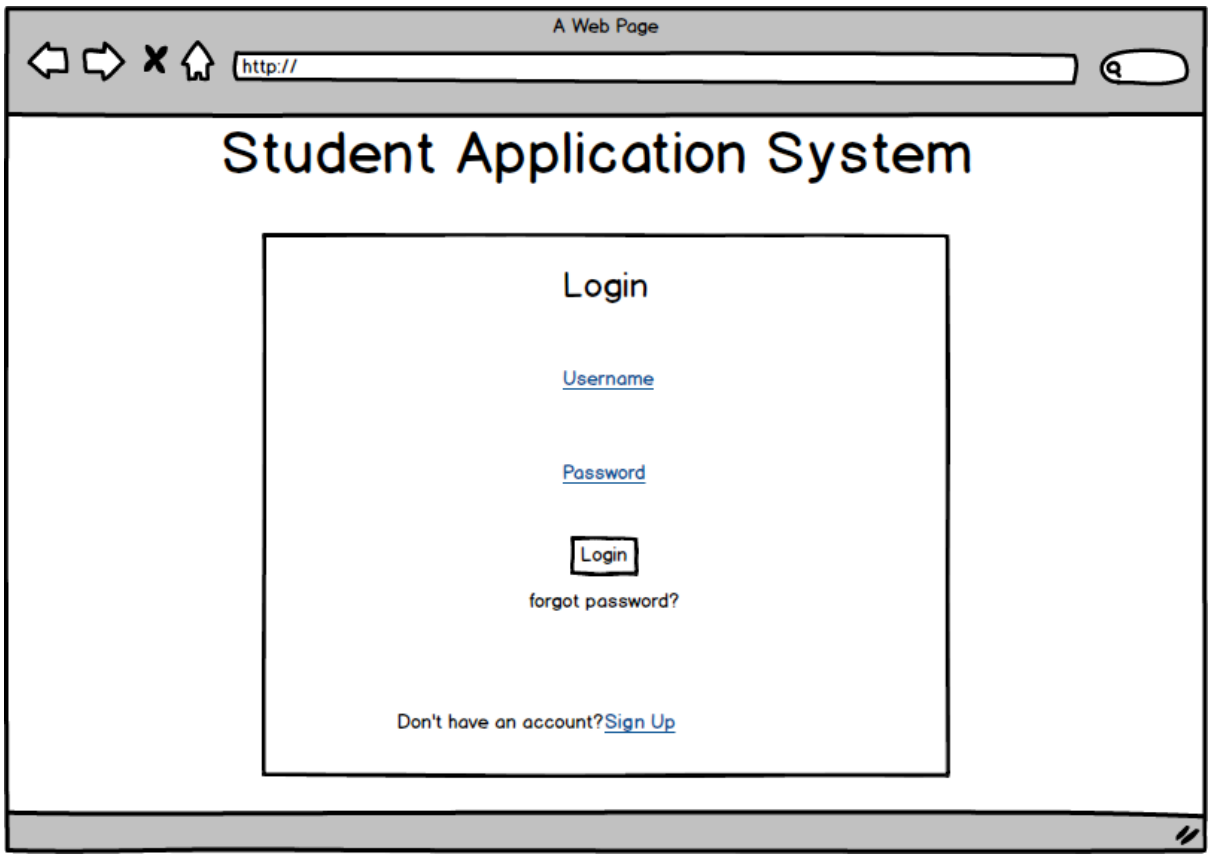
Picture 11: SASAdmin Page



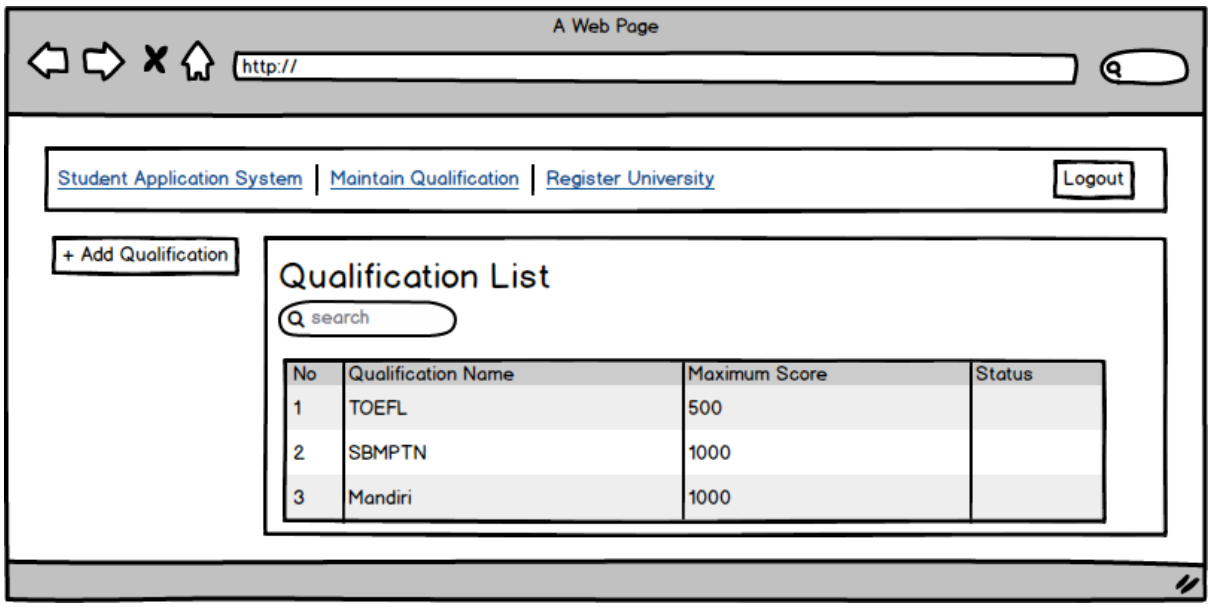
Picture 12: Applicant Page



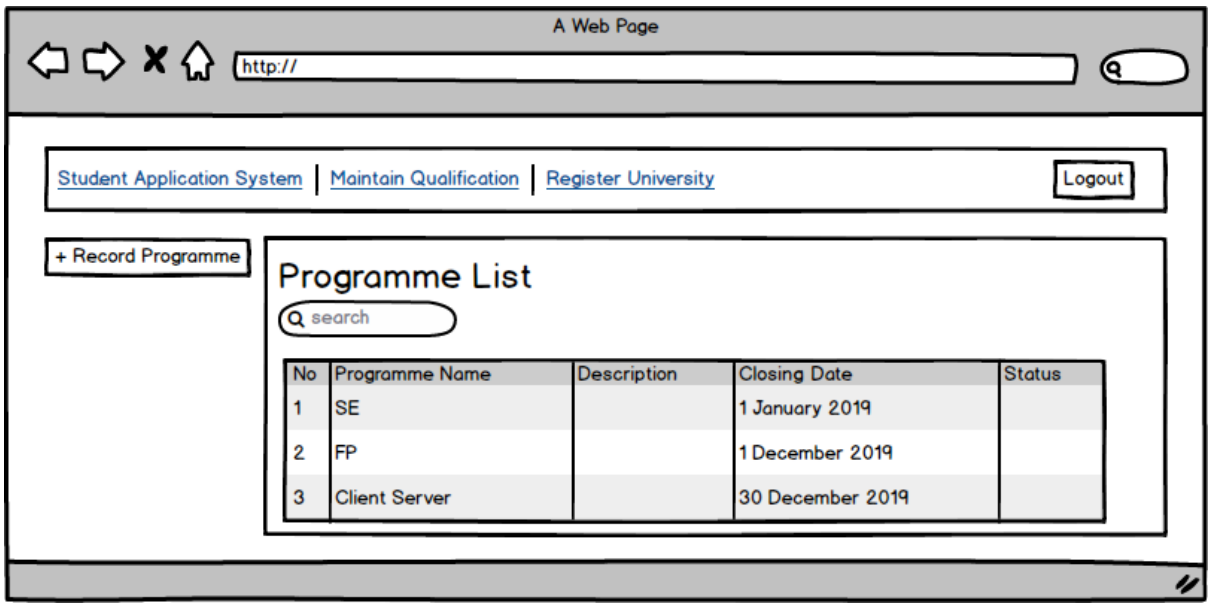
Picture 13: Change Status Page



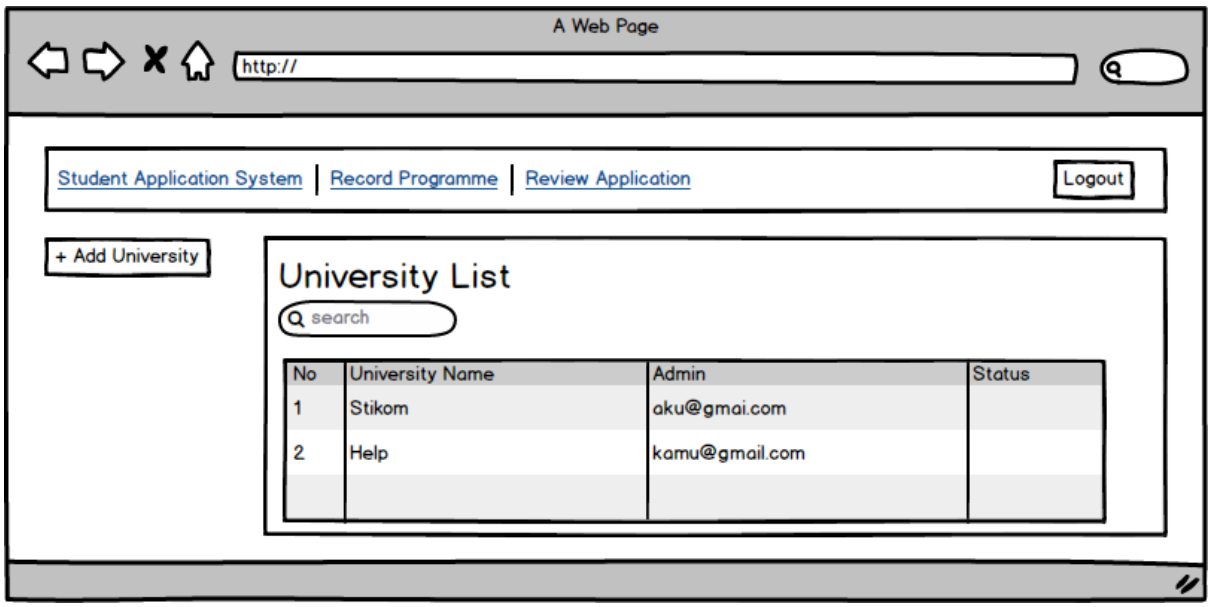
Picture 14: Home Page



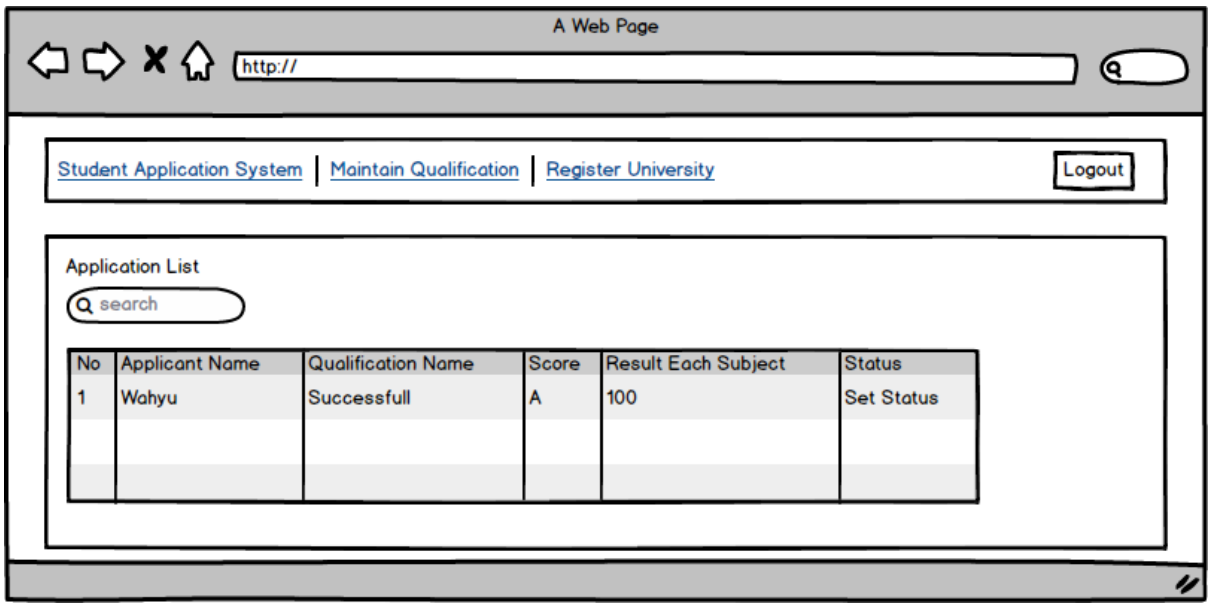
Picture 15: Maintain Qualification Page



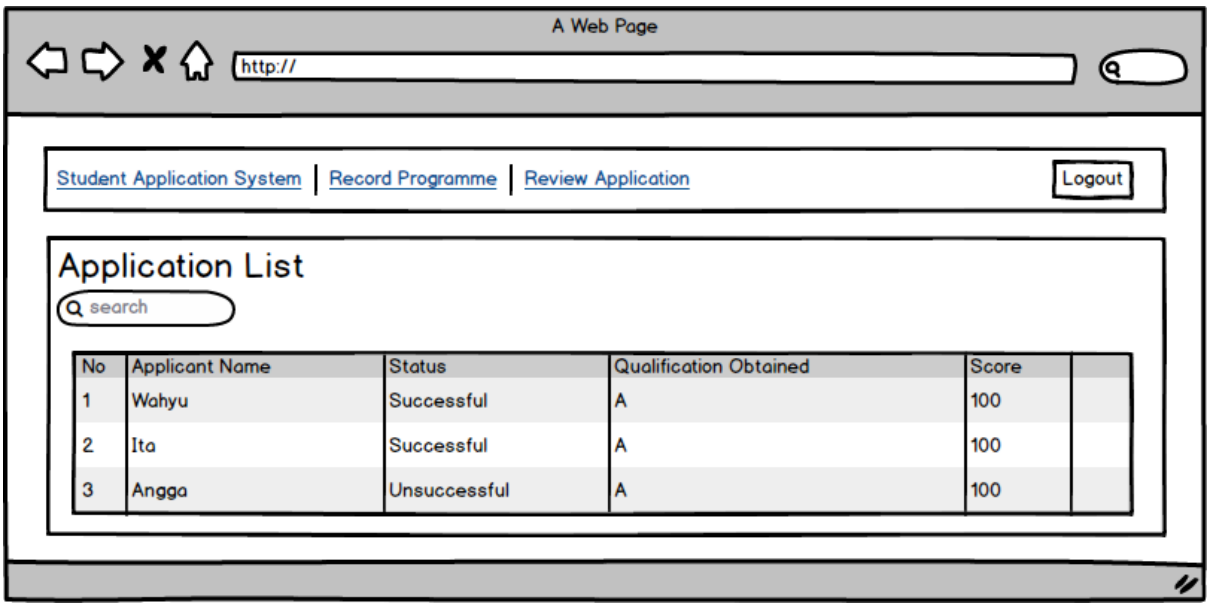
Picture 16: Record Programme Page



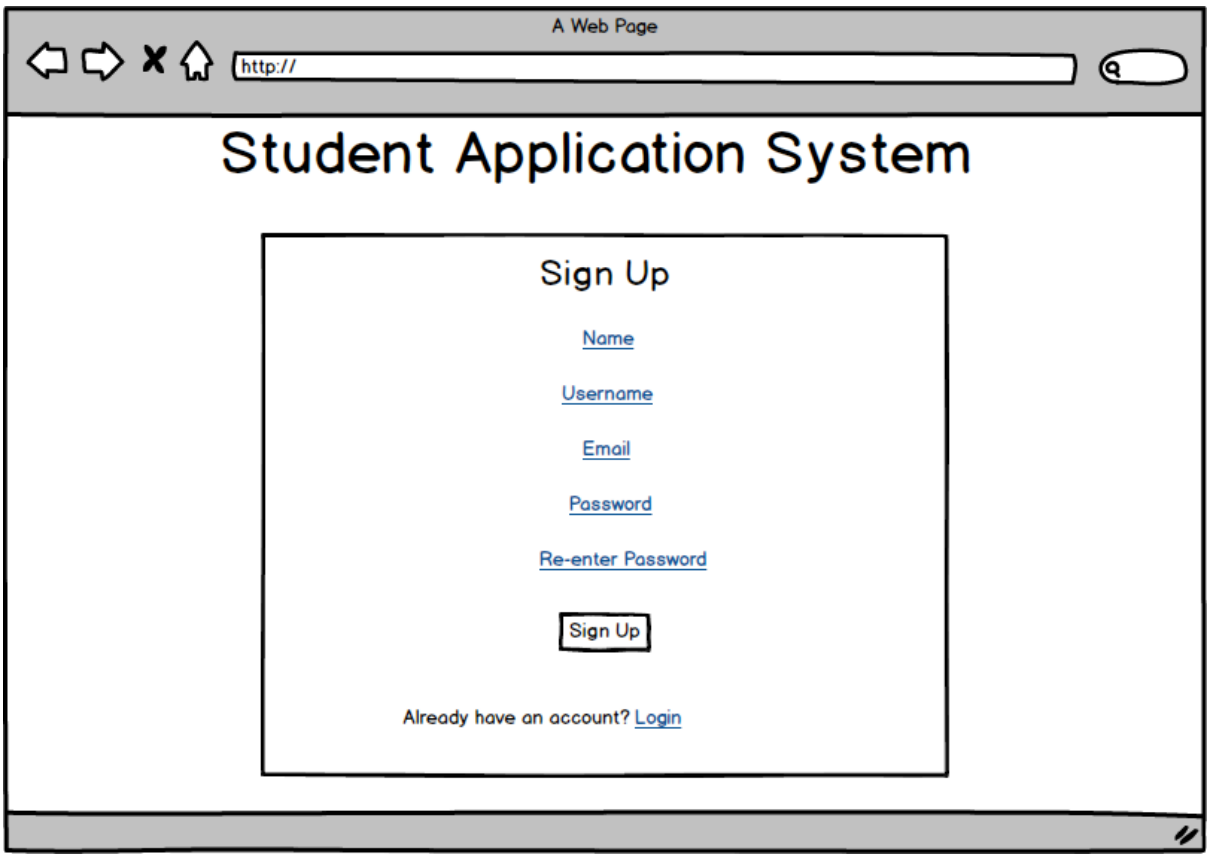
Picture 17: Register University Page



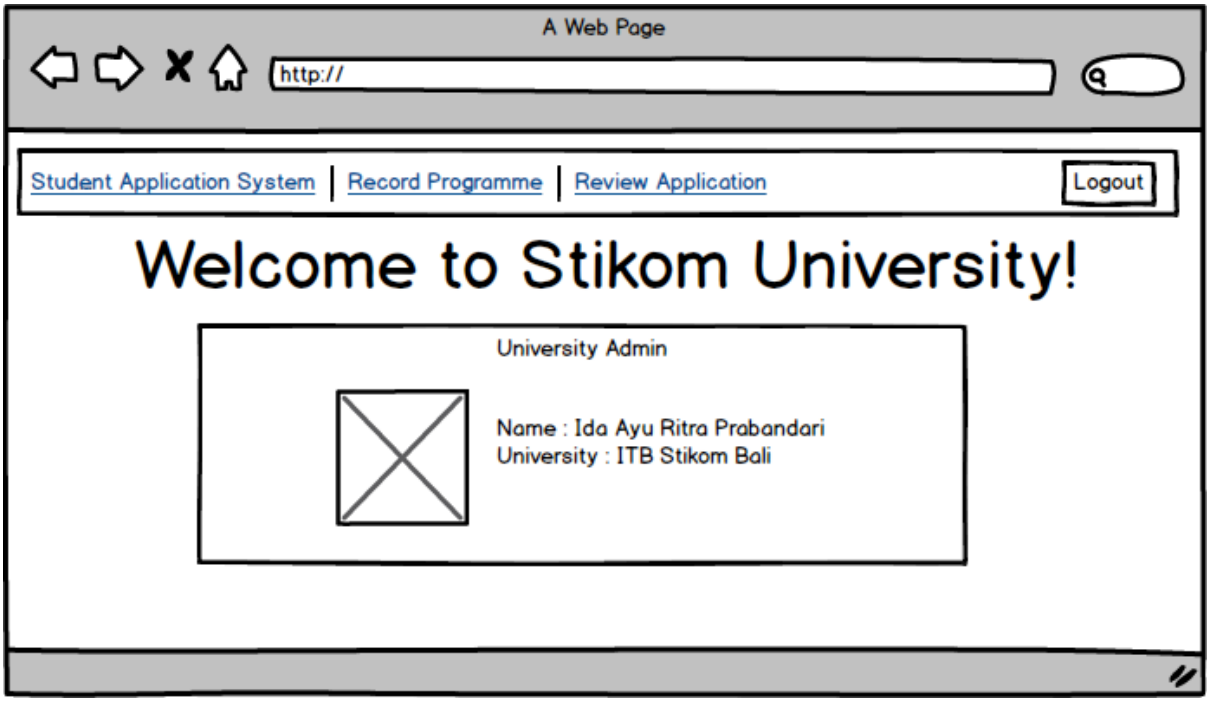
Picture 18: Review Application Page



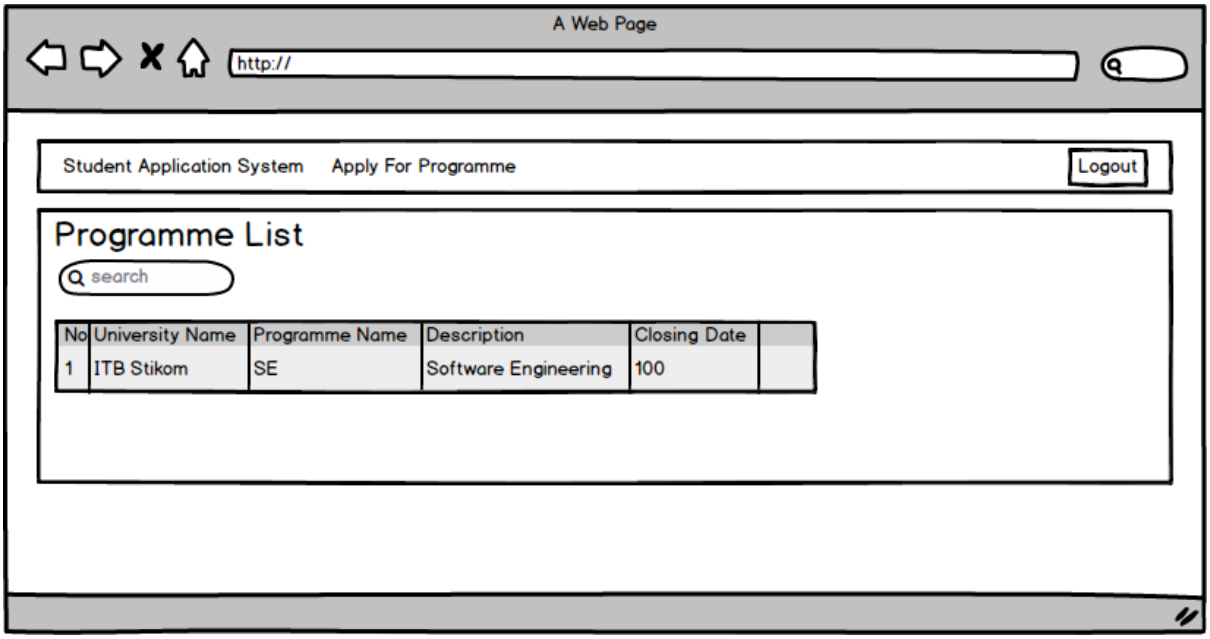
Picture 19: Select Application Page



Picture 20: Sign Up Page



Picture 21: University Admin Page



Picture 22: Apply For Programme

Iteration 1

Use Cases

Use Case	Develop By	Testing
SetUp Qualification	Ritra	Iteration 1
Register University	Ritra	Iteration 2
Record Programme	Wahyu	Iteration 1
Apply For Programme	Wahyu	Iteration 2
Review Application	Wahyu	Iteration 2

Test Objectives

In developing a system, as humans we are not free from mistakes. Errors can appear in the code and design during the development phase. Thus, the main purpose of testing is to find bugs and errors that might occur when developing software. Testing is important to build trust and comfort in the quality of the software before sharing the software with clients. This stage is very important to ensure the requirements are met and all functions work well, to maintain client reliability and satisfaction. Testing is also important to prevent errors that occur in the future, because it will be more difficult and more expensive to repair.

For this Student Application System, our first test goal is to ensure validation of entry functions and password data are successfully encrypted. Next, we want to test whether the application (web page) has the same appearance in various browsers. From the perspective of SAS Admin and University Admin, it must be able to add, edit and delete data. Meanwhile, the Applicant must be able to see data only related to the Applicant, and can send messages to the SAS Admin and University Admin. We also need to test whether all links function properly and point to the right page.

Test Plan

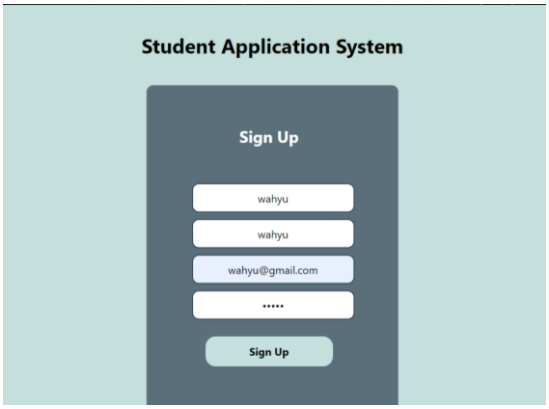
In this Student Application System, we will test the functions that will be carried out by SAS Admins & University Admins as well as functions that will be carried out by Applicants. We want to make sure the system limits are functioning. For example, the Applicant cannot edit or delete data, except edit the profile itself. Applicants should not be able to add anything, except add a message to ask the admin questions. Login must also be tested, so that the Applicant after logging in is not directed to the SAS Admin or University Admin

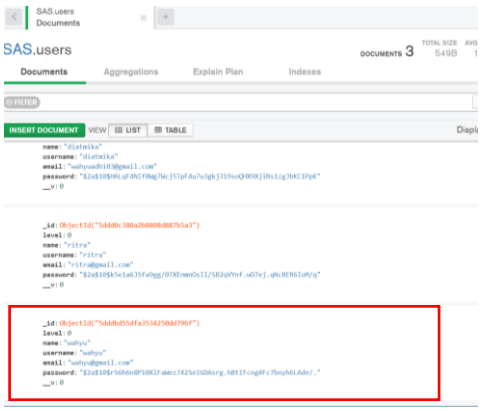
homepage, and SAS Admin & University Admin is not directed to the Applicant home page. During the testing process, we will test the function 3 to 5 times, to verify that the function has worked perfectly.

Unit Testing

1. Register Applicant

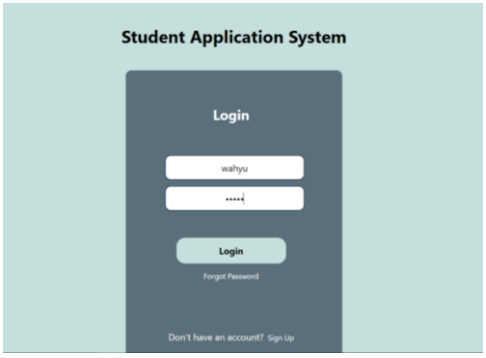
   This test will test the Applicant in creating a new account that will be used to login to the system. After logging in the Applicant will immediately be directed to the Applicant's home menu.

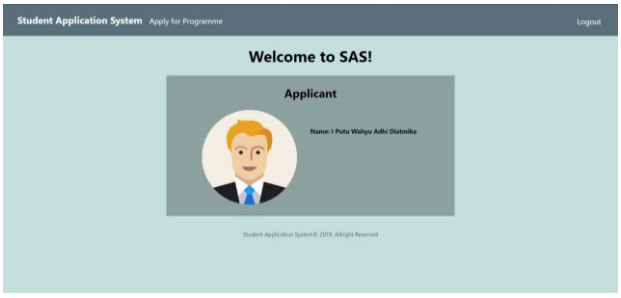




2. Login Applicant

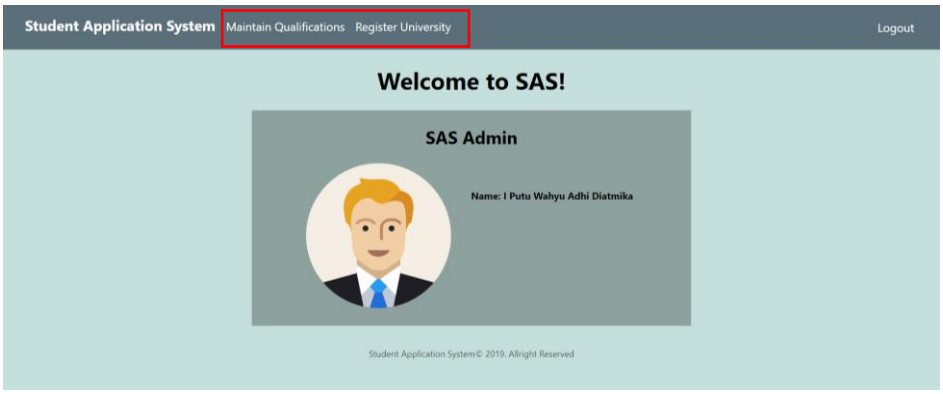
   In this test will show the Applicant in logging into the system by entering the username and password that was made before.

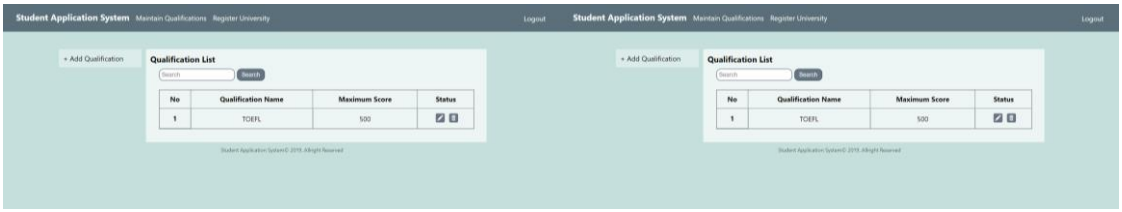




3. SAS Admin

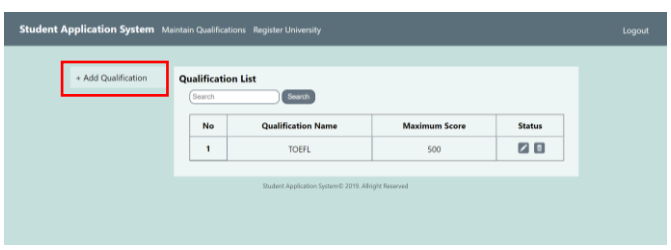
This test will test the SAS Admin home menu which has the function to direct the page to Maintain Qualification and University Register.

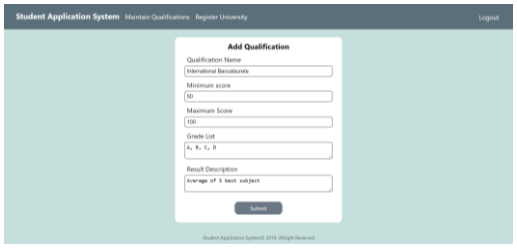


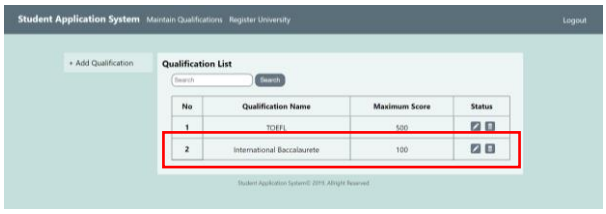


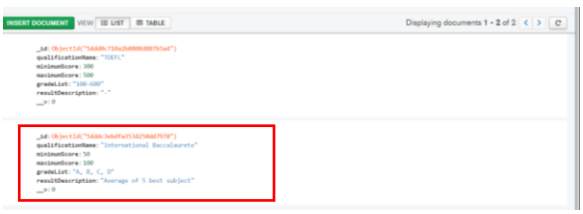
4. Maintain Qualification

Pada testing ini akan mengetes fungsi add qualification yang digunakan untuk menambahkan qualification





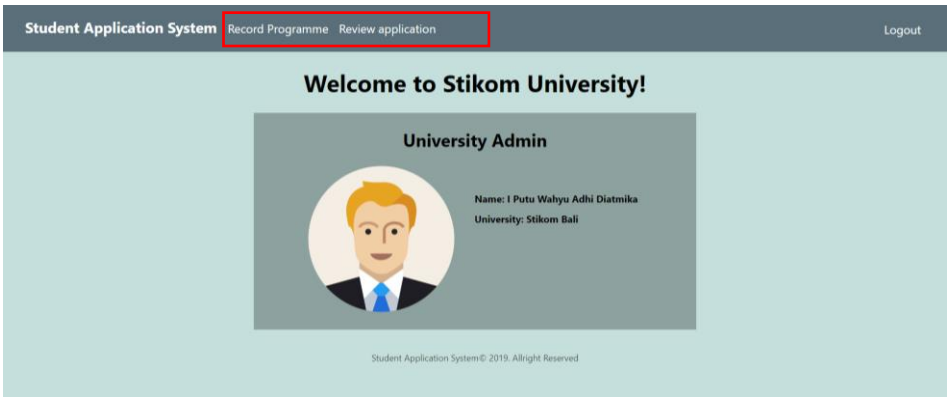


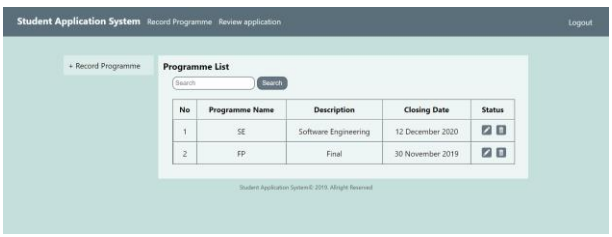


The data input on the add qualification was successfully added to the database and the data contained in the table was in accordance with the database.

5. University Admin

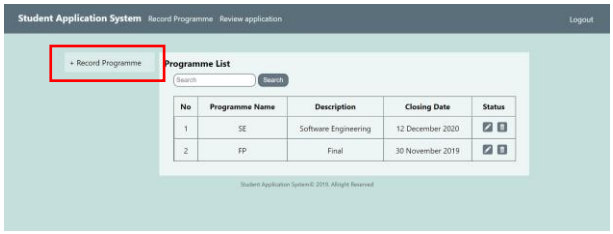
In this test will test the University Admin home menu which has a function to direct the page to the Record Programme and Review Applications.

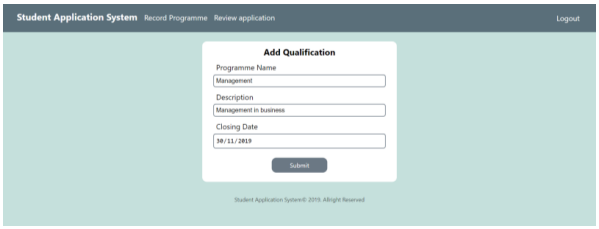


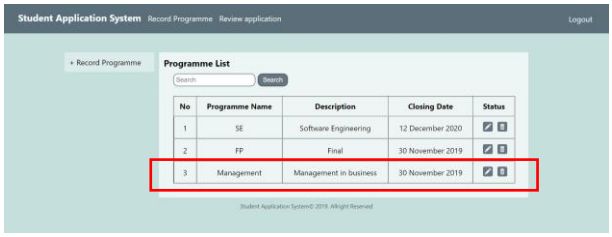


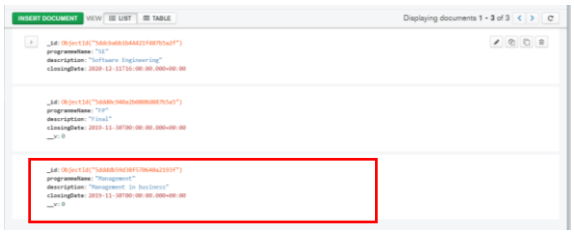
6. Record Programme

In this test, we will test the record programme function which is used to add the program to university.





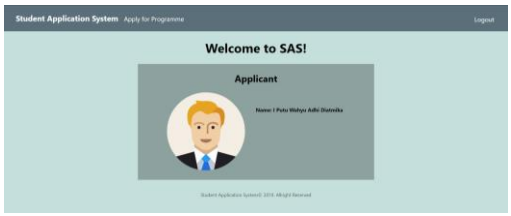


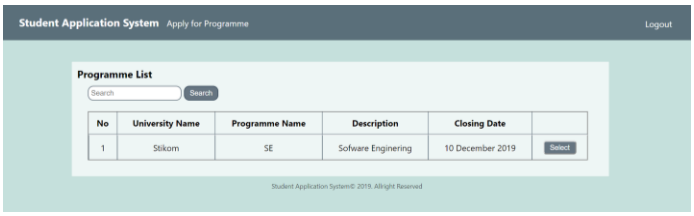


The data input in the program record was successfully added to the database and the data contained in the table was in accordance with the database.

7. Applicant

In this test, we will test the Applicant home menu, which has the function to direct the page to Apply for Program.





Integration Testing

In this integration testing, the testing will use the Selenium-WebDriver application. The system will run based on the functions that exist in each use case. The results of testing will be exported into JavaScript files.

1. Register Applicant

```
// Generated by Selenium IDE
const { Builder, By, Key, until } = require('selenium-webdriver')
const assert = require('assert')

describe('Register', function() {
  this.timeout(30000)
  let driver
  let vars
  beforeEach(async function() {
    driver = await new Builder().forBrowser('chrome').build()
    vars = {}
  })
  afterEach(async function() {
    await driver.quit();
  })
  it('Register', async function() {
    await driver.get("http://localhost:3000/signUp")
    await driver.setRect(516, 680)
    await driver.findElement(By.name("name")).click()
    await driver.findElement(By.name("name")).sendKeys("wahyuadhi")
    await driver.findElement(By.name("username")).click()
    await driver.findElement(By.name("username")).sendKeys("wahyuadhi")
    await driver.findElement(By.name("email")).click()
    await driver.findElement(By.name("email")).sendKeys("wahyu@gmail.com")
    await driver.findElement(By.name("email")).click()
    await driver.findElement(By.name("email")).sendKeys("wahyuadhi@gmail.com")
    await driver.findElement(By.name("password")).click()
    await driver.findElement(By.name("password")).sendKeys("wahyuadhi")
    await driver.findElement(By.css("input:nth-child(1)")).click()
  })
})
```

Log Reference

8. type on name=email with value wahyu@gmail.com OK
9. click on name=email OK
10. type on name=email with value wahyuadhi@gmail.com OK
11. click on name=password OK
12. type on name=password with value wahyuadhi OK
13. click on css=input:nth-child(1) OK

'Register' completed successfully


```
username: "ritra"
email: "ritra@gmail.com"
password: "$2a$10$k5e1a6J5fa9gg/0TXEnmnOsI1/SB2qVYnf.wD7ej.qNcBER6IoM/q"
__v: 0

_id: ObjectId("5dddbd55dfa3534250dd796f")
level: 0
name: "wahyu"
username: "wahyu"
email: "wahyu@gmail.com"
password: "$2a$10$rS6h6n8P10XlFaWez742SeIGDAsrg.hBtIFcng4Fc7bnyh6LAde/."
__v: 0

_id: ObjectId("5ddde0e0d38f570640a21940")
level: 0
name: "wahyuadhi"
username: "wahyuadhi"
email: "wahyuadhi@gmail.com"
password: "$2a$10$C4o7jZEPppUkVdcM9HLHs3eOT.UmB2mKAsAatbwAT21PC2E3JUQQvO"
__v: 0
```

2. Login

```
// Generated by Selenium IDE
const { Builder, By, Key, until } = require('selenium-webdriver')
const assert = require('assert')

describe('Login', function() {
  this.timeout(30000)
  let driver
  let vars
  beforeEach(async function() {
    driver = await new Builder().forBrowser('chrome').build()
    vars = {}
  })
  afterEach(async function() {
    await driver.quit();
  })
  it('Login', async function() {
    await driver.get("http://localhost:3000/")
    await driver.setRect(516, 680)
    await driver.findElement(By.name("username")).sendKeys("wahyuadhi")
    await driver.findElement(By.css(".login")).click()
    await driver.findElement(By.name("password")).click()
    await driver.findElement(By.name("password")).sendKeys("wahyuadhi")
    await driver.findElement(By.css("input:nth-child(1)")).click()
  })
})
```

Log Reference

2. setWindowSize on 516x680 OK
3. type on name=username with value wahyuadhi OK
4. click on css=.login OK
5. click on name=password OK
6. type on name=password with value wahyuadhi OK
7. click on css=input:nth-child(1) OK

'Login' completed successfully

3. Maintain Qualification

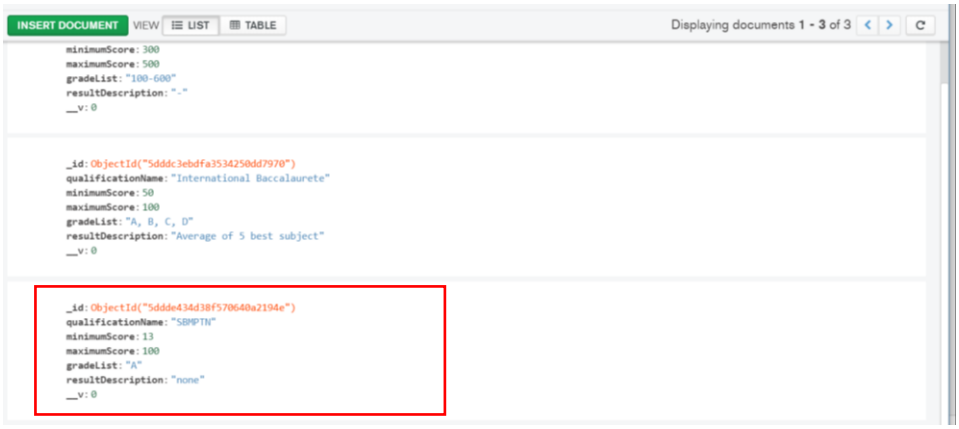
```
// Generated by Selenium IDE
const { Builder, By, Key, until } = require('selenium-webdriver')
const assert = require('assert')

describe('Maintain', function() {
  this.timeout(30000)
  let driver
  let vars
  beforeEach(async function() {
    driver = await new Builder().forBrowser('chrome').build()
    vars = {}
  })
  afterEach(async function() {
    await driver.quit();
  })
  it('Maintain', async function() {
    await driver.get("http://localhost:3000/maintainQualification")
    await driver.setRect(516, 680)
    await driver.findElement(By.linkText("+ Add Qualification")).click()
    await driver.findElement(By.id("qualificationName")).click()
    await driver.findElement(By.id("qualificationName")).sendKeys("SBMPTN")
    await driver.findElement(By.id("minimumScore")).click()
    await driver.findElement(By.id("minimumScore")).sendKeys("13")
    await driver.findElement(By.id("maximumScore")).click()
    await driver.findElement(By.id("maximumScore")).sendKeys("100")
    await driver.findElement(By.id("gradeList")).click()
    await driver.findElement(By.id("gradeList")).sendKeys("A")
    await driver.findElement(By.id("resultDescription")).click()
    await driver.findElement(By.id("resultDescription")).sendKeys("none")
    await driver.findElement(By.css(".submitbutton")).click()
  })
})
```

Log Reference

9. type on id=maximumScore with value 100 OK
10. click on id=gradeList OK
11. type on id=gradeList with value A OK
12. click on id=resultDescription OK
13. type on id=resultDescription with value none OK
14. click on css=.submitbutton OK

'Maintain' completed successfully



4. Record Programme

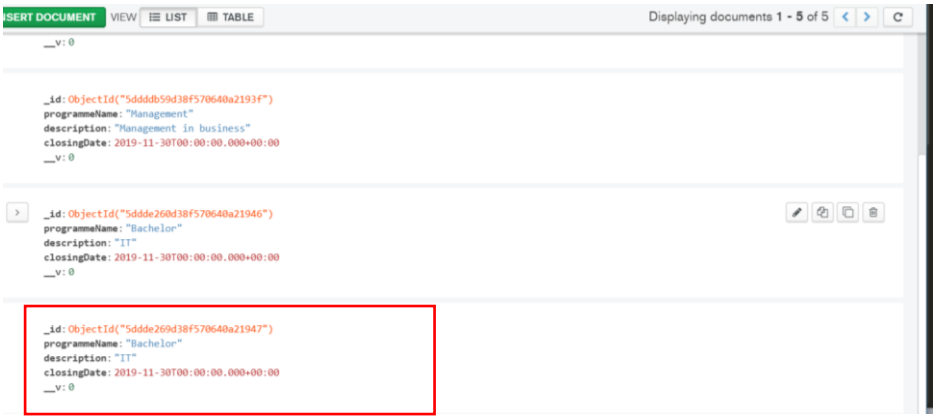
```
// Generated by Selenium IDE
const { Builder, By, Key, until } = require('selenium-webdriver')
const assert = require('assert')

describe('Record Programme', function() {
  this.timeout(30000)
  let driver
  let vars
  beforeEach(async function() {
    driver = await new Builder().forBrowser('chrome').build()
    vars = {}
  })
  afterEach(async function() {
    await driver.quit();
  })
  it('Record Programme', async function() {
    await driver.get("http://localhost:3000/recordProgramme")
    await driver.setRect(516, 680)
    await driver.findElement(By.linkText("+ Record Programme")).click()
    await driver.findElement(By.id("programmeName")).click()
    await driver.findElement(By.id("programmeName")).sendKeys("Bachelor")
    await driver.findElement(By.id("description")).click()
    await driver.findElement(By.id("description")).sendKeys("IT")
    await driver.findElement(By.id("closingDate")).click()
    await driver.findElement(By.id("closingDate")).sendKeys("2019-11-30")
    await driver.findElement(By.css(".submitbutton")).click()
  })
})
```

Log Reference

5. type on id=programmeName with value Bachelor OK
6. click on id=description OK
7. type on id=description with value IT OK
8. click on id=closingDate OK
9. type on id=closingDate with value 2019-11-30 OK
10. click on css=.submitbutton OK

'Record Programme' completed successfully

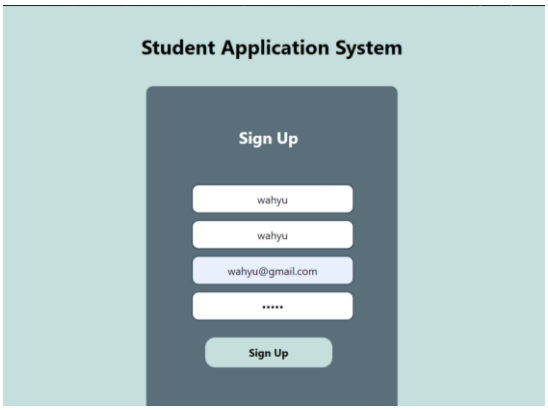


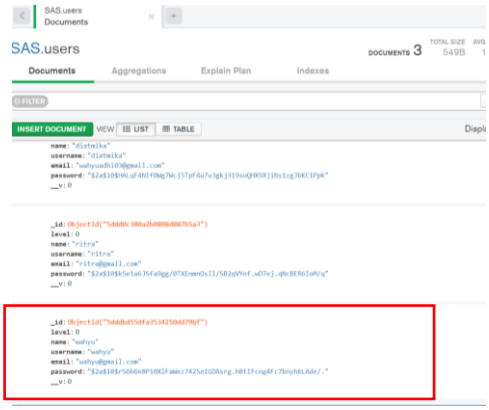
System Testing

1. Functional Requirements

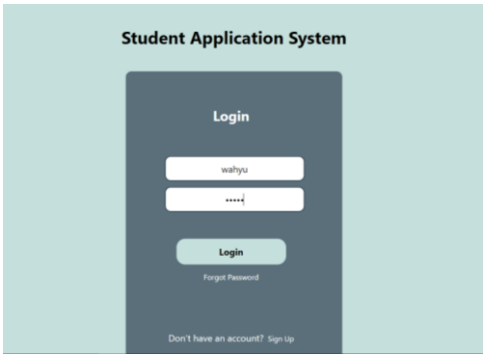
At this stage, the system will carry out testing in accordance with the functional requirements that have been determined previously. The following shows the results of testing that has been done.

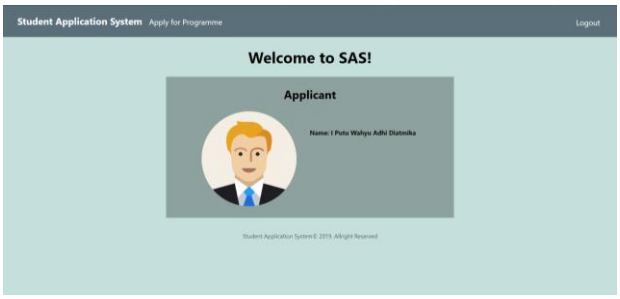
a. Register Applicant



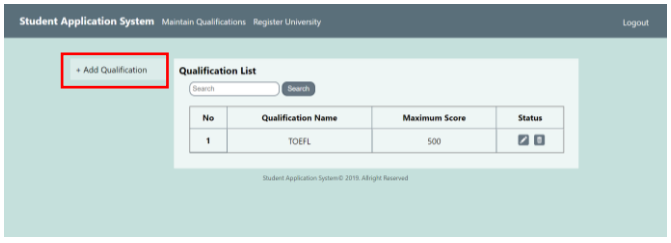


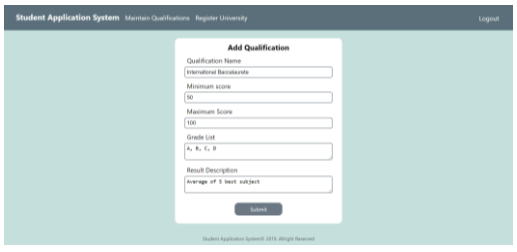
b. Login Applicant

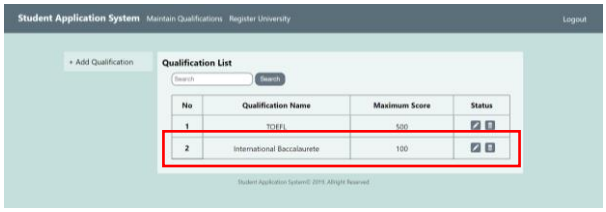
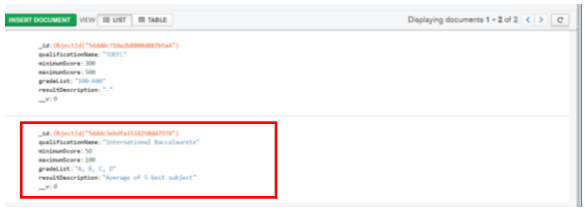




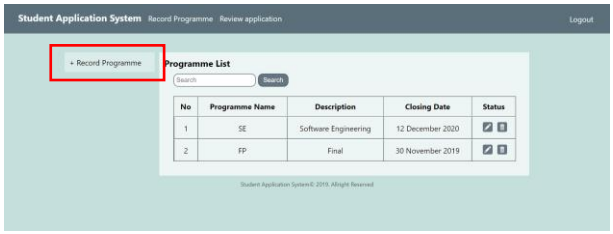
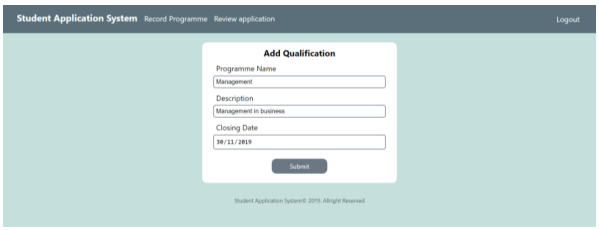
c. Maintain Qualification

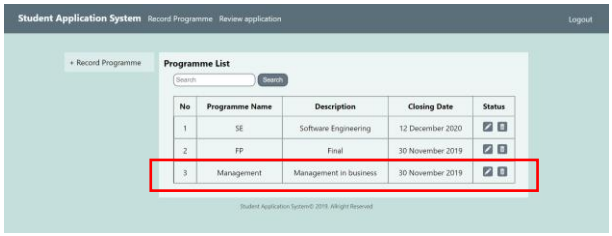
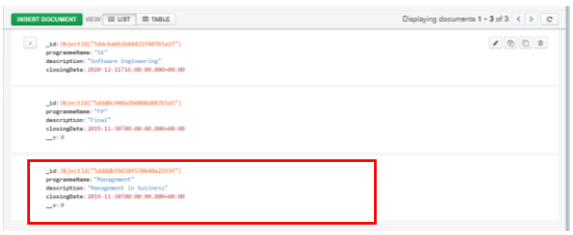




d. Record Programme

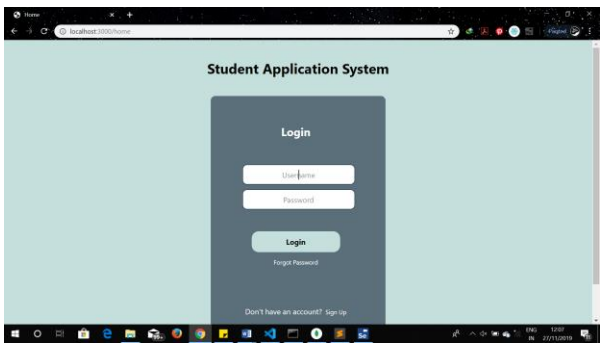
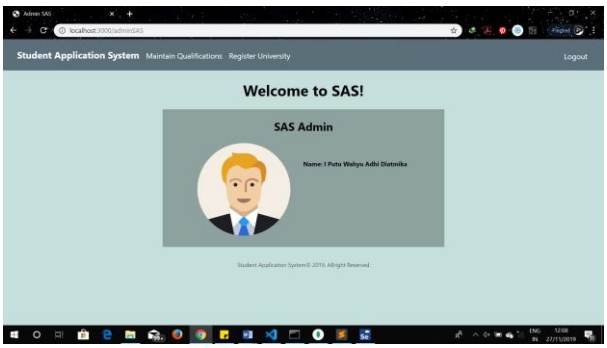



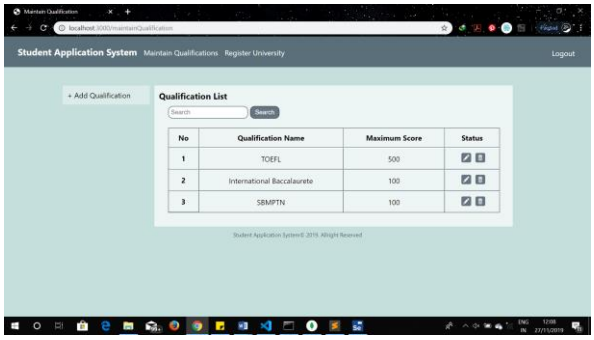
2. Non-functional

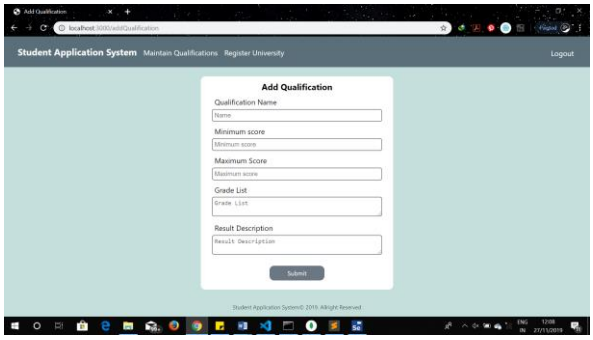
At this stage, the system will carry out testing according to the non-functional requirements that have been determined previously. The following shows the results of testing that has been done.

a. Usability

The system already have a simple interface design and easy to use also easy to understand.

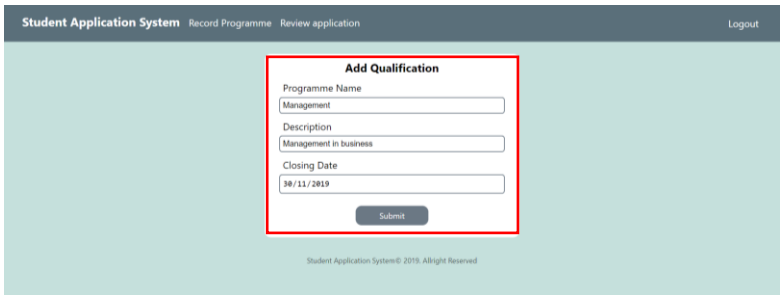



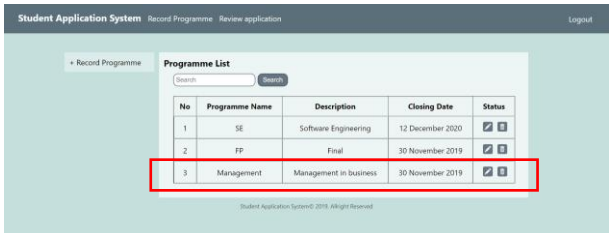


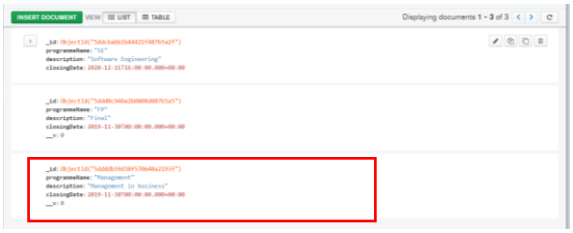


b. Accuracy and Precision

The data entered by the user has been appropriate, accurate, and precise and there are no errors. And also, the data is in accordance with the data contained in the database.







Test Analysis Report

From a series of tests we conducted regarding unit testing, integration testing, and system testing, it can be analyzed that the system developed is functioning properly because it has met the desired requirements. In this testing phase, the use cases tested are Login, Register, Maintain Qualifications, and Record Program.

In unit testing for use cases Maintain Qualification is tested on the Add Qualification and Display List Qualification functions. In the unit testing for use case the Record Program is tested on the Add Program and Display List Program functions. For the Integration Testing stage, all functions in each use case are tested using a predetermined application, Selenium-WebDriver. And, the System Testing is tested by functional and non-functional matching.

Some functional tests that have been tested are:

- Register Applicant :

  This page is only intended for Applicants, for creating new accounts.

- Login :

  This page functions to enter the system, but you must register beforehand if you do not have an account.

- Maintain Qualification :

  This page functions to add and view a list of qualifications that have been determined.

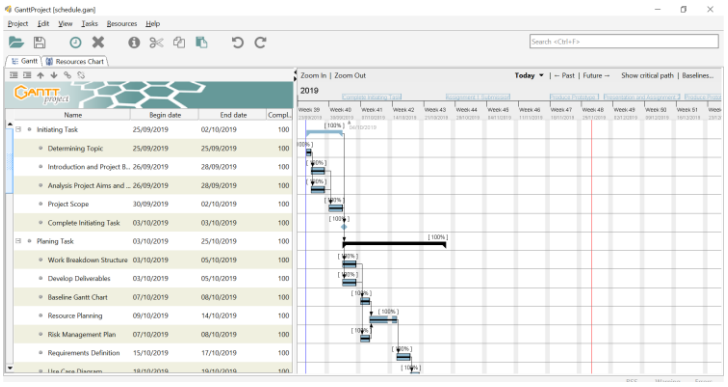
- Record Programme :

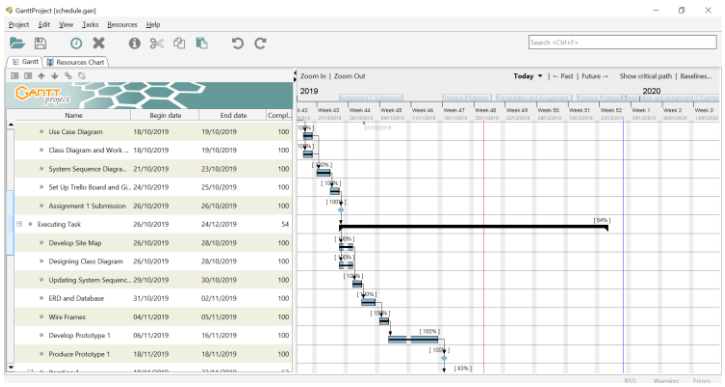
  This page functions is to add and view a list of programs that have been determined.

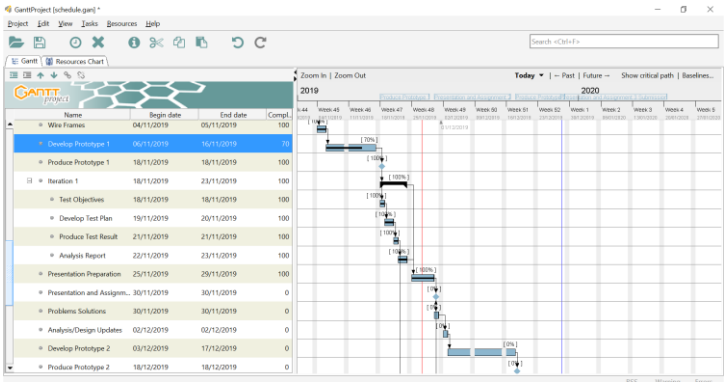
In Non-functional testing, for Usability the system already has a simple interface design and easy to use also easy to understand and for the Accuracy and Precision the data entered by the user has been appropriate, accurate, and precise and there are no errors. And also, the data is in accordance with the data contained in the database.

Updated Gantt Chart

Gantt charts are used to set a schedule in accordance with a predetermined task. It can be seen in the image below the Gantt chart that has been going well, but in developing prototype 1 we have not been able to meet the standards that we have set previously.

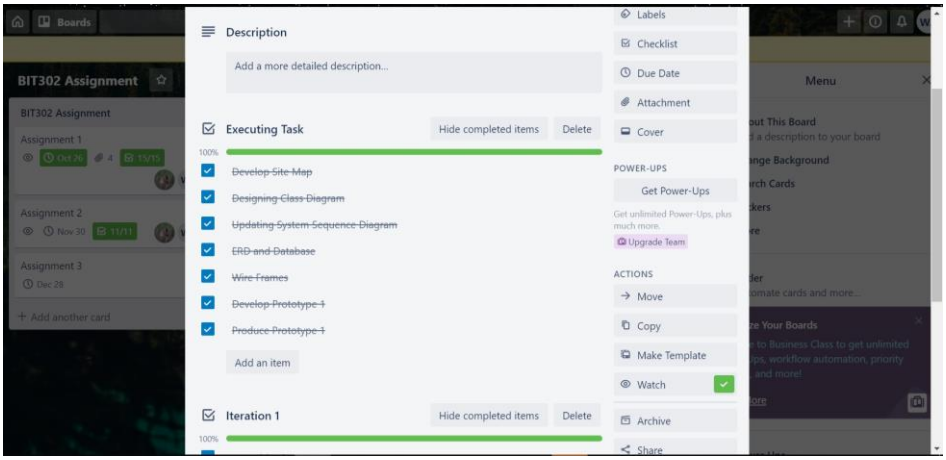






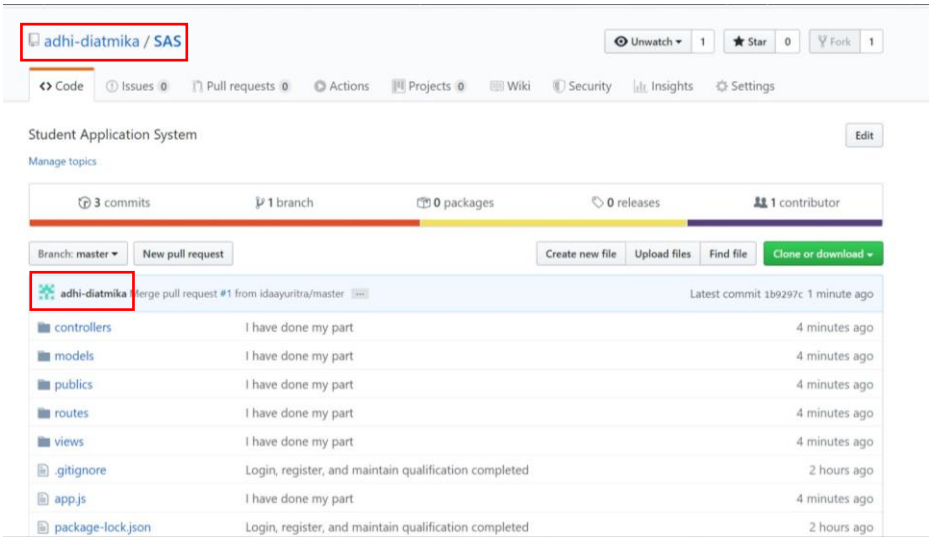
Updated Trello

Trello is used to monitor every task that has been completed by the team in working on the project. In the picture below it can be seen that each task can be completed properly.

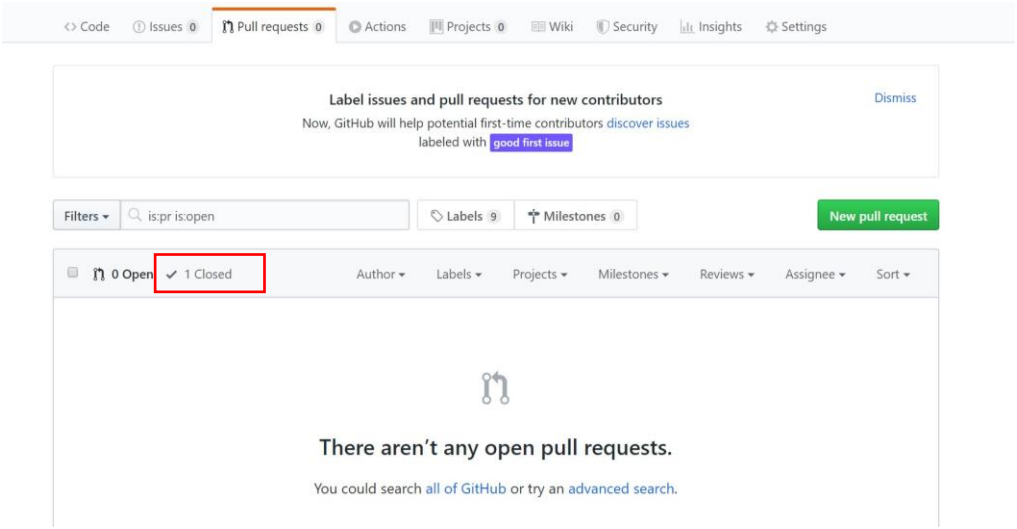


Collaboration via GitHub

GitHub is used to combine systems that have been completed by each team member. In the picture below, it can be seen that the team leader has set up the repository and has done the push function to update to the repository.



In the picture below shows that a team member has done the pull function to make a request so that the system that has been completed can be combined with the previous file. It can be seen in the red city that there is 1 pull request and the request has been closed.



In the picture below it can be seen that the team leader has performed a merge function to merge all files that have been completed by the team members