

# Team Report

## Introduction

This LIBRUS 2.0 program aims to help education groups to manage members, determine their progression during university studies; and show what they can view or edit depending on which roles they play in the university.

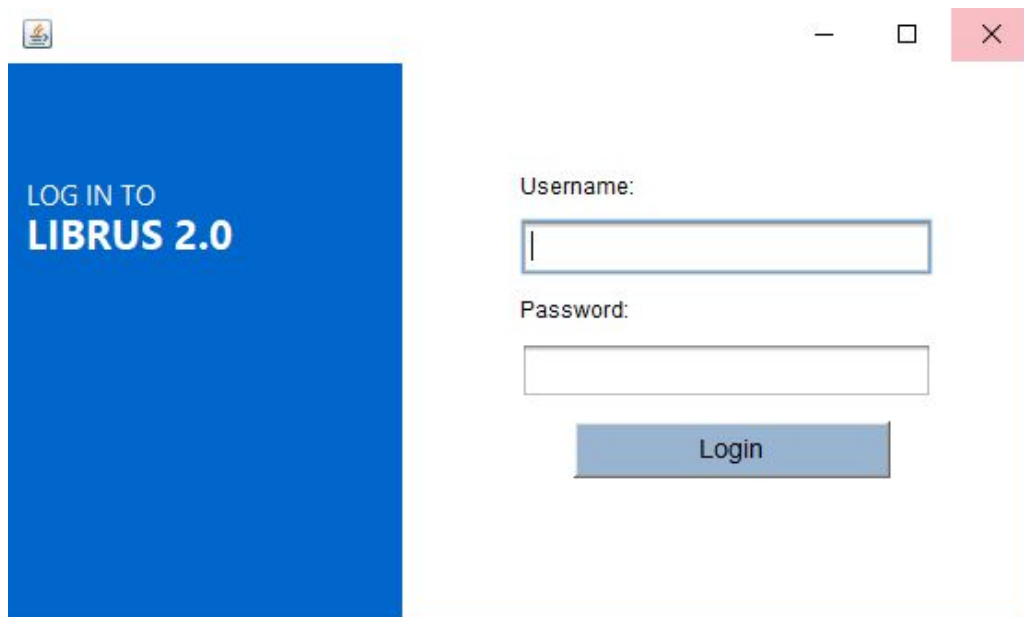


IMAGE1. Program Login Page

To meet the requirements for designing the system needs, our group first overviewed the instructions. We then divided our UML works into four parts - Database, Use Case, Class, and State Machine. We designed the diagrams for each part and assigned tasks for every teammate at the very beginning of the project with Draw.IO. Each

individual diagram shows relationships between elements the project needed and the program was designed based on the connections.

We found that it is necessary to start by establishing a database for creating information tables and storing data that is needed by different types of users (Admin, Registrar, Teacher, Student). Beyond the establishment, we are working on the features each operation should display, and designing their class files based on the use case and class diagrams.

```
CREATE TABLE ModuleDegree (  
    codeOfModule varchar(7) NOT NULL,  
    codeOfDegree varchar(6) NOT NULL,  
    level char NOT NULL,  
    isCore bit NOT NULL,  
    FOREIGN KEY (codeOfDegree) REFERENCES Degree(codeOfDegree),  
    FOREIGN KEY (codeOfModule) REFERENCES Module(codeOfModule)  
);  
  
CREATE TABLE Student (  
    registrationNum int NOT NULL AUTO_INCREMENT,  
    codeOfDegree varchar(6),  
    username varchar(255) NOT NULL,  
    title varchar(2),  
    surname varchar(255),  
    forname varchar(255),  
    email varchar(255),  
    personalTutor varchar(255),  
    level char,  
    PRIMARY KEY (codeOfModule)  
    FOREIGN KEY (codeOfDegree) REFERENCES Degree(codeOfDegree),  
    FOREIGN KEY (username) REFERENCES Users(username)  
);
```

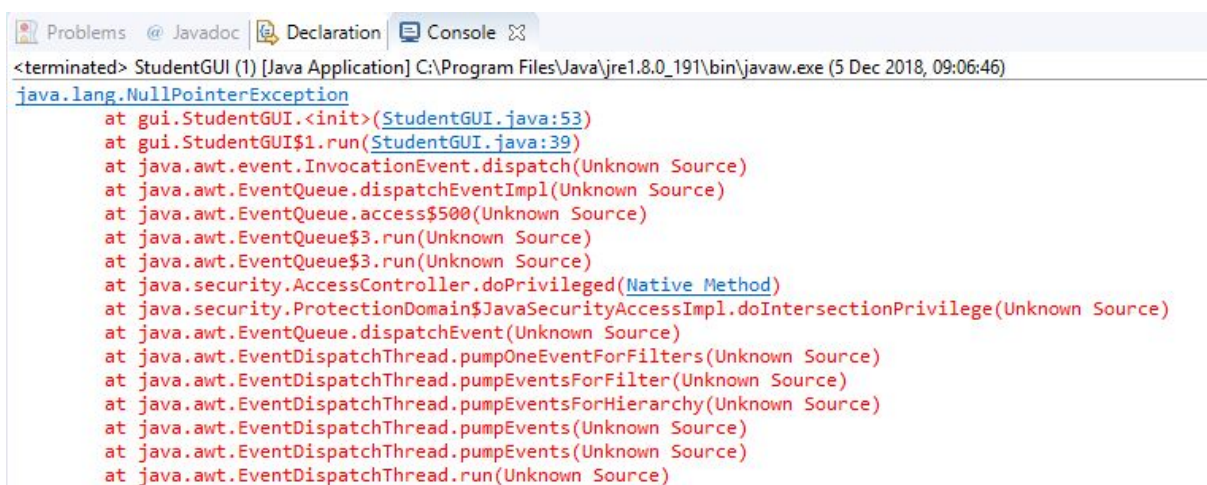
IMAGE2. Some examples of database tables

Designing the user interface is the next task we dealt with. The program was followed by the state machine diagram which made it

clear to design. The program is comprehensible to different types of users with clear user distinctions, so every user is restricted to enter their own pages. Graphic design was also an important role in the project - we created a smooth, clear and streamlined style which is more functional. Images in the Best Features part below show the differences between them.

This system used BCrypt as our security hashing function according to its resistance to various dangers as it is one of the most well-known and safe anti-hacking tools. The details of why we used BCrypt as a hashing function will be discussed in the Security Features part.

After everyone finished their individual tasks, the complete system was connected part by part through a database and tested by the group to make sure it worked perfectly. The process of debugging is our last step of this project.

A screenshot of a Java IDE's console window. The title bar shows 'Problems', 'Javadoc', 'Declaration', and 'Console'. The console text starts with '<terminated> StudentGUI (1) [Java Application] C:\Program Files\Java\jre1.8.0\_191\bin\javaw.exe (5 Dec 2018, 09:06:46)'. Below this, a red line indicates a 'java.lang.NullPointerException'. The stack trace follows, listing various Java methods and their line numbers, such as 'at gui.StudentGUI.<init>(StudentGUI.java:53)', 'at gui.StudentGUI\$1.run(StudentGUI.java:39)', and 'at java.awt.EventQueue.access\$500(Unknown Source)'. The stack trace ends with 'at java.awt.EventQueue.run(Unknown Source)'.

```
<terminated> StudentGUI (1) [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (5 Dec 2018, 09:06:46)
java.lang.NullPointerException
    at gui.StudentGUI.<init>(StudentGUI.java:53)
    at gui.StudentGUI$1.run(StudentGUI.java:39)
    at java.awt.event.InvocationEvent.dispatch(Unknown Source)
    at java.awt.EventQueue.dispatchEventImpl(Unknown Source)
    at java.awt.EventQueue.access$500(Unknown Source)
    at java.awt.EventQueue$3.run(Unknown Source)
    at java.awt.EventQueue$3.run(Unknown Source)
    at java.security.AccessController.doPrivileged(Native Method)
    at java.security.ProtectionDomain$JavaSecurityAccessImpl.doIntersectionPrivilege(Unknown Source)
    at java.awt.EventQueue.dispatchEvent(Unknown Source)
    at java.awt.EventDispatchThread.pumpOneEventForFilters(Unknown Source)
    at java.awt.EventDispatchThread.pumpEventsForFilter(Unknown Source)
    at java.awt.EventDispatchThread.pumpEventsForHierarchy(Unknown Source)
    at java.awt.EventDispatchThread.pumpEvents(Unknown Source)
    at java.awt.EventDispatchThread.pumpEvents(Unknown Source)
    at java.awt.EventDispatchThread.run(Unknown Source)
```

IMAGE3. ERROR message met during testing

Different situations and data were tested and problems were fixed one by one. Finally the program was finished and the latest version of the system was completed.

# UML Use Case Diagram

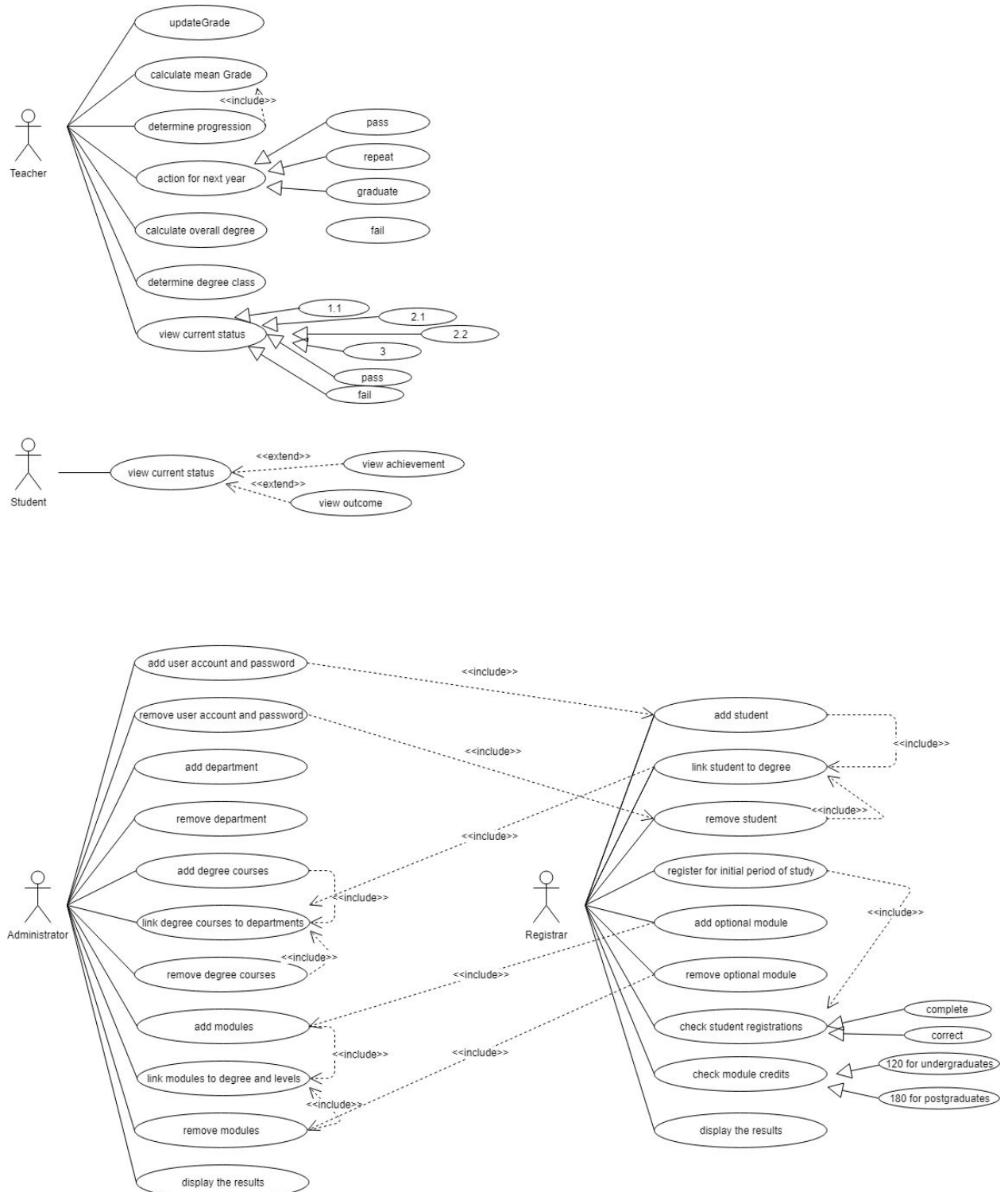


IMAGE4. Use Case UML Diagram

# UML Class Diagram

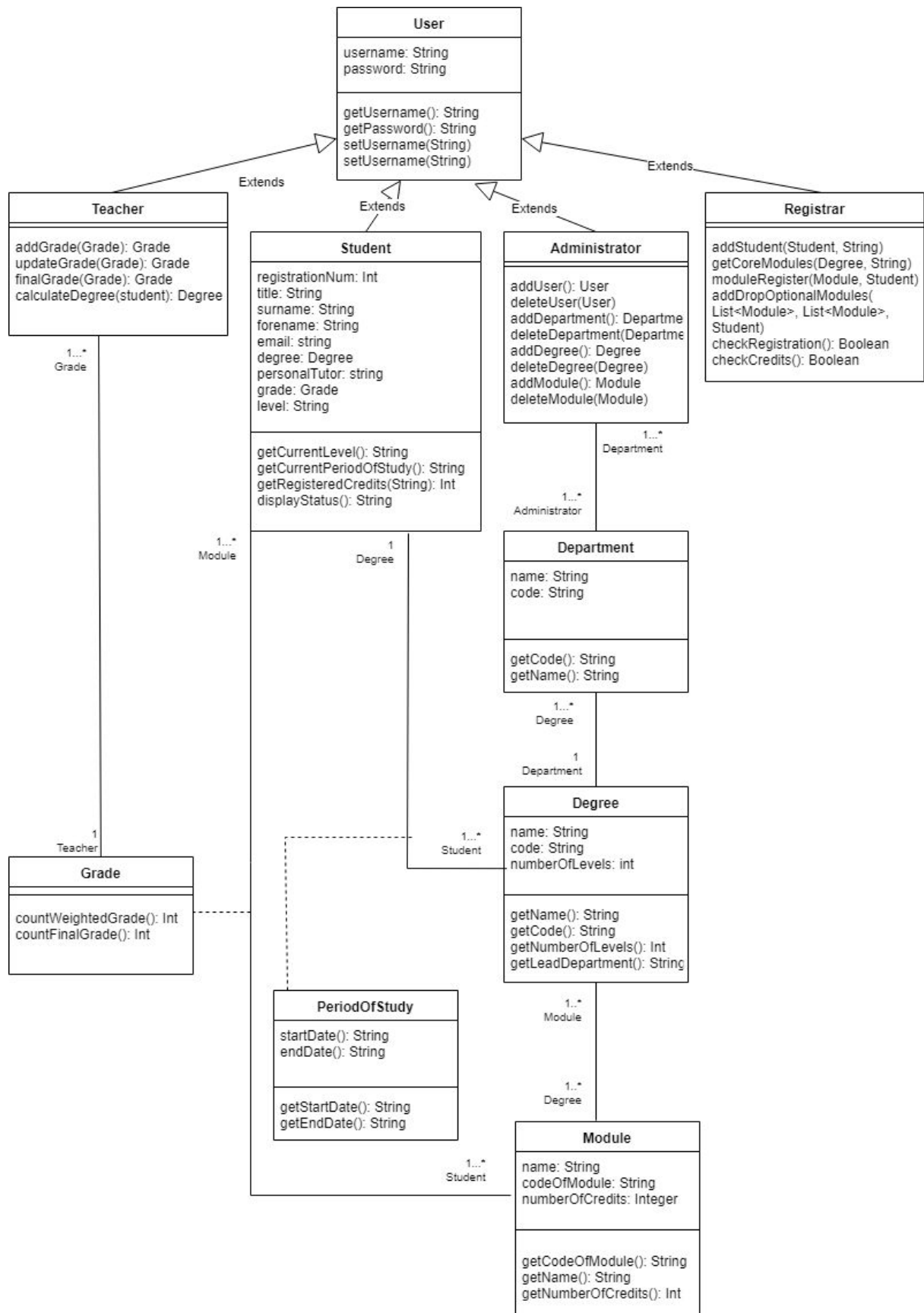


IMAGE5. Class UML Diagram

# UML Class Diagram of the Normalised Database Model

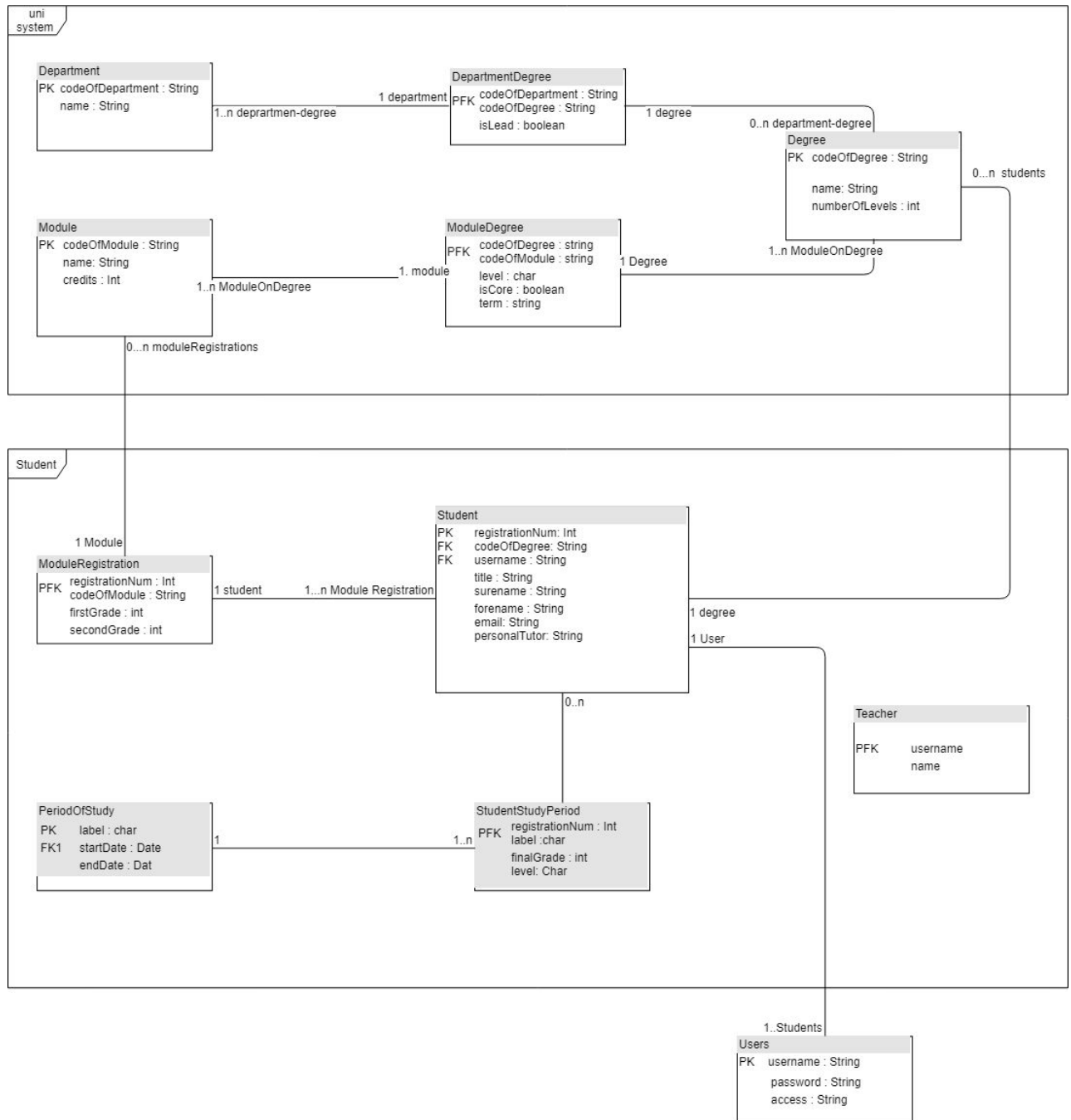


IMAGE6.. Normalised Database UML Model

## UML State Machine Diagram

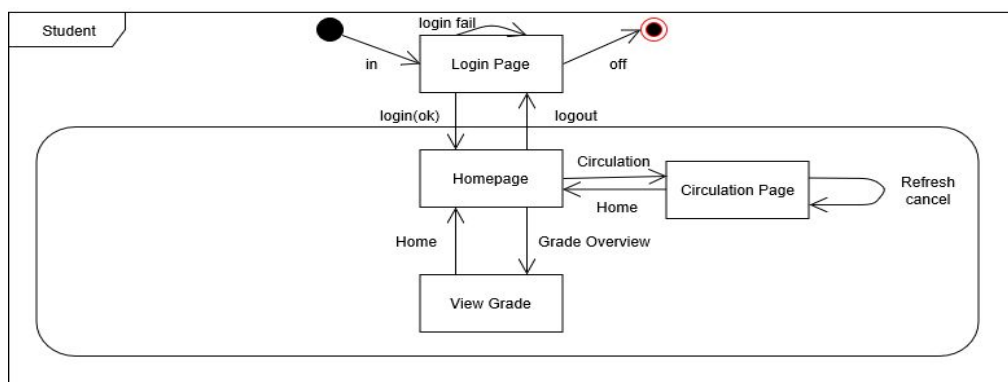
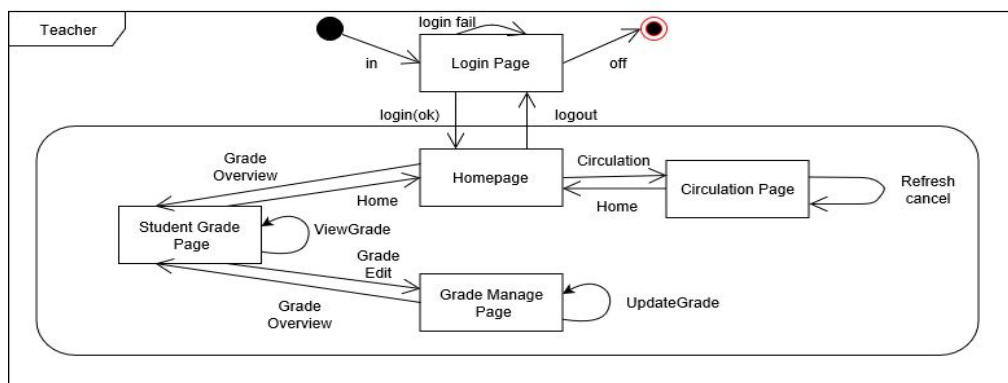
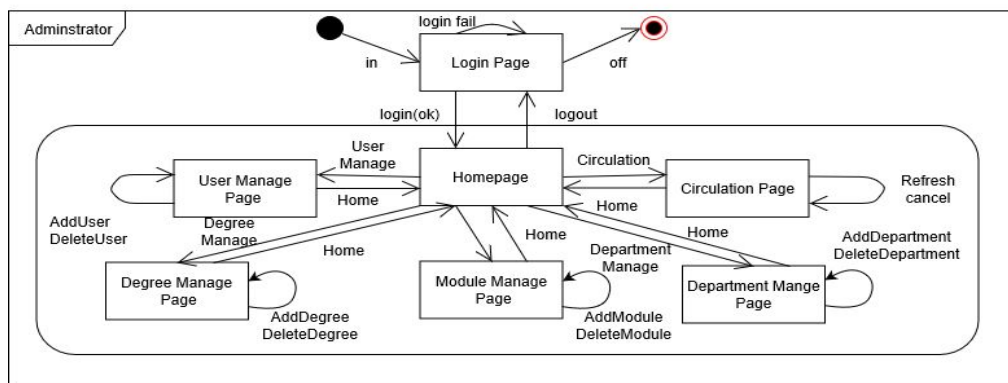
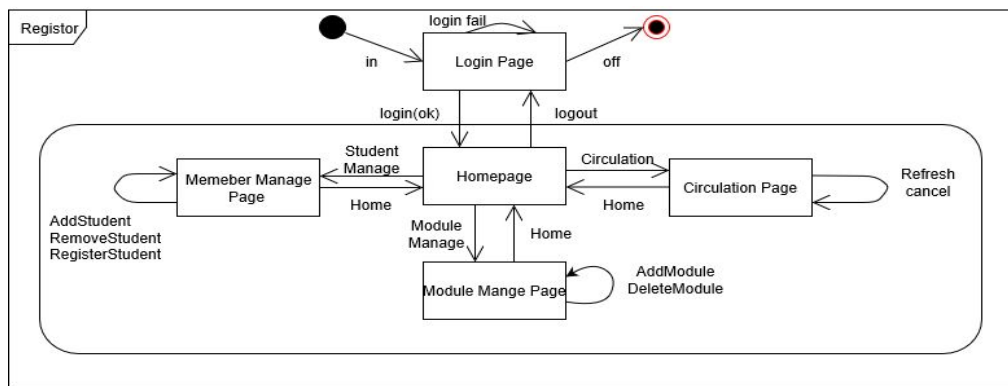


IMAGE7. State Machine UML Diagram

## Screenshots of Best Aspects

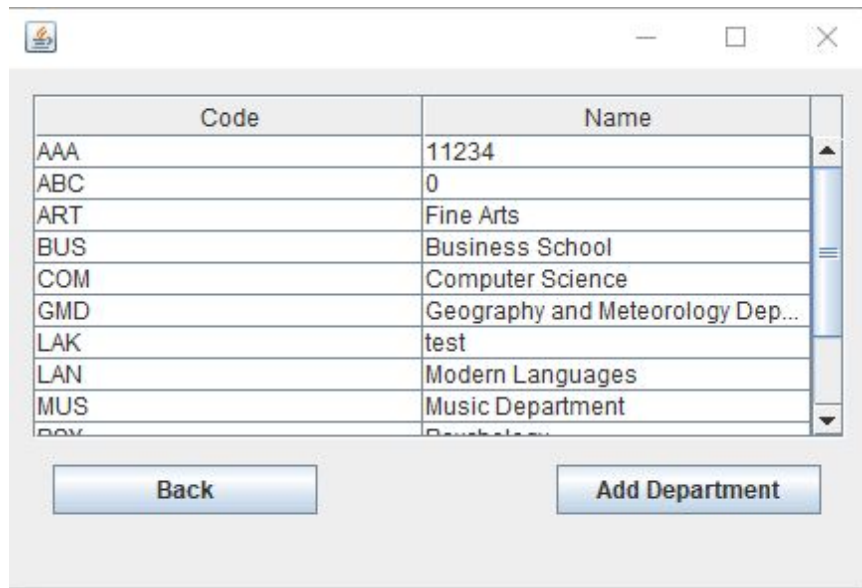


IMAGE8. Older Version (DepartmentGUI)

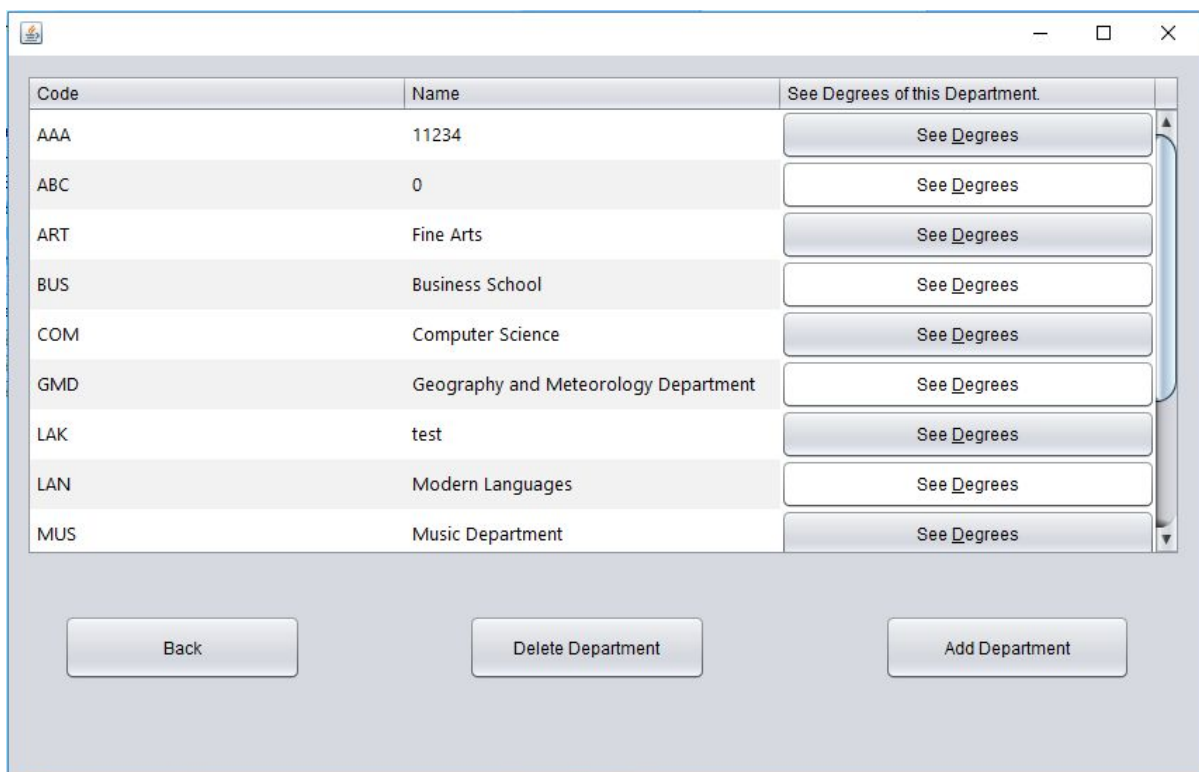


IMAGE9. Current Version(DepartmentGUI 2.0)



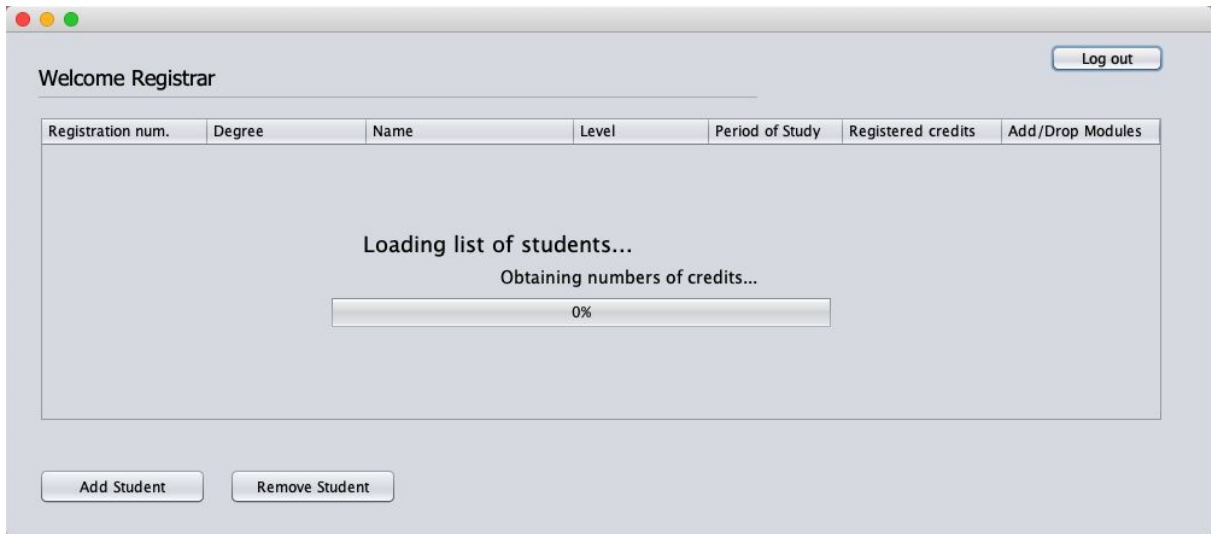


IMAGE10. Registrar Page Loading Start Screen

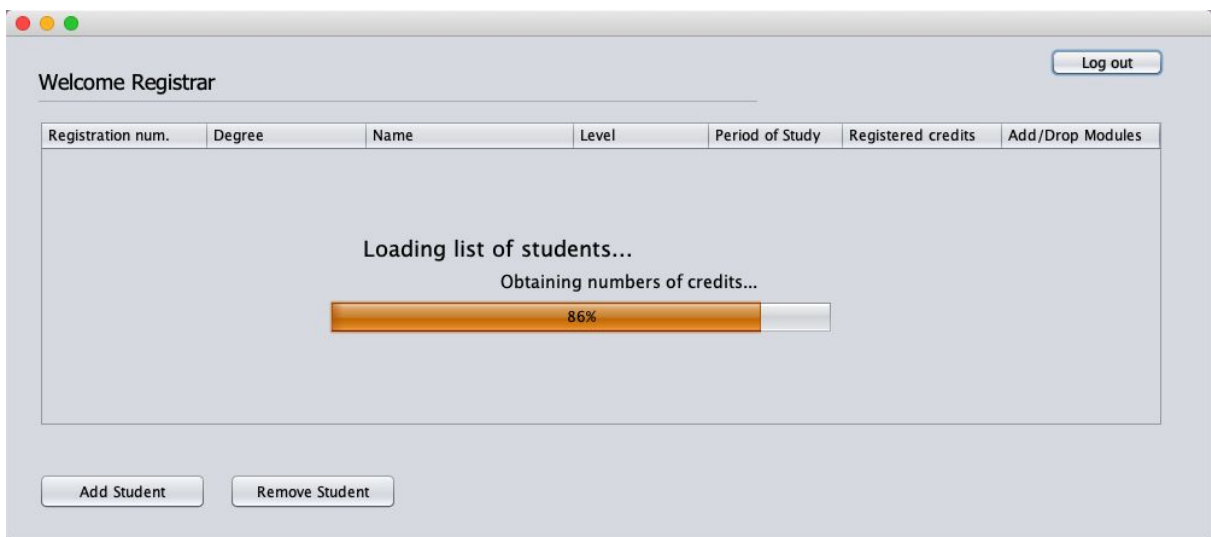


IMAGE11. Registrar Page Loading Screen

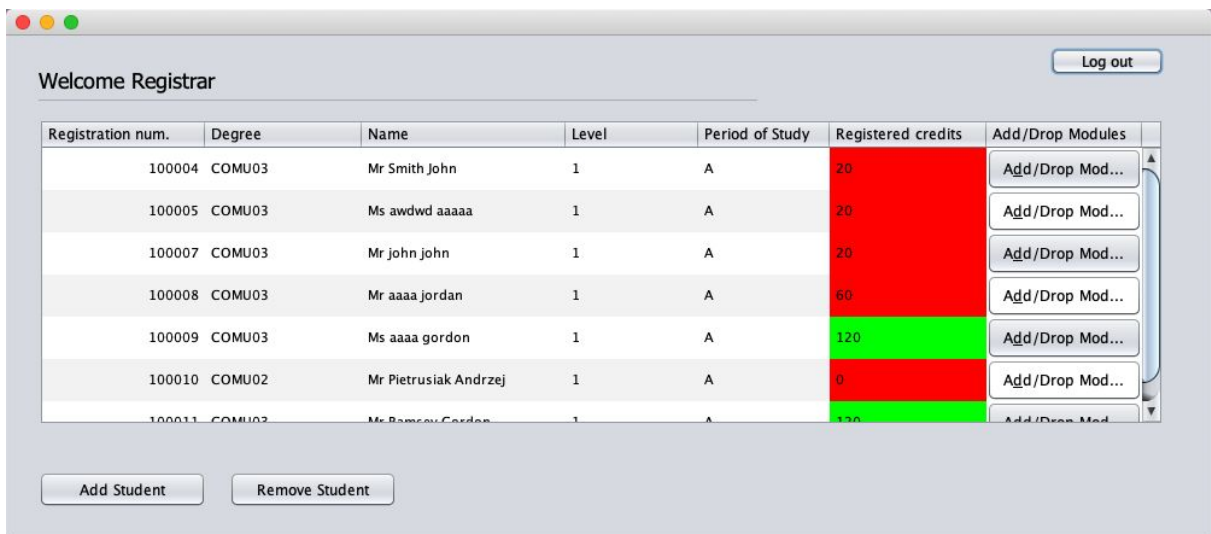


IMAGE12. Registrar Page Loading End Screen

## Security Features

Security has been an important influencing factor in the the overall status of the system. In order to get access to the other pages, the user should provide valid information. To keep the password safe, we use BCrypt to hash the password in order to make sure every password has its own hashing algorithm which makes the server more secure. BCrypt does not only do plain text but also compromises other websites for users. Hackers have found solutions around this system as the algorithm used is not a direct one-way option in theory. The only way hackers can succeed is to guess passwords until they get the access authorization. It also adds a long string of bytes to the passwords which also strongly protect the system, which is good against 'Rainbow Table'. The image below shows how BCrypt secures the passwords and make it really difficult for hackers to hack.

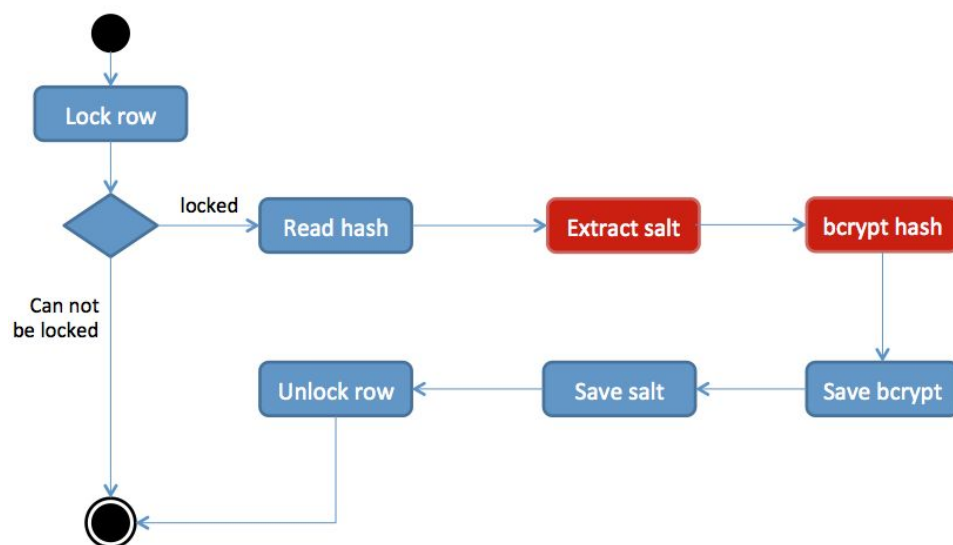


IMAGE13. Algorithm of changing Password using BCrypt(Dr. Christian Winkler, 2013)

Additionally, if different users registered with the same password, they are able to find the other owners of that password. To deal with this problem, 'salting' would be an advantage, which provides

additional encryption. Bcrypt also executed iterations logarithmically and updatably. It makes the CPU work more intensively and makes the system more secure. The updatability makes it easier to deal with brute force attacks by verifying passwords and increasing their lengths, including generating a new hash with a higher number of iterations.

Apart from these factors, storing a hashed password in MD5, for example, is not as secure as storing directly with BCrypt. Some of the entropy is overlapped by the original algorithm. Moreover, it is still tremendously more secure than storing the password normally hashed. As one user logs in, a pure version of his password is saved into the database and the manager still can change the flag so the password verifications only use BCrypt.

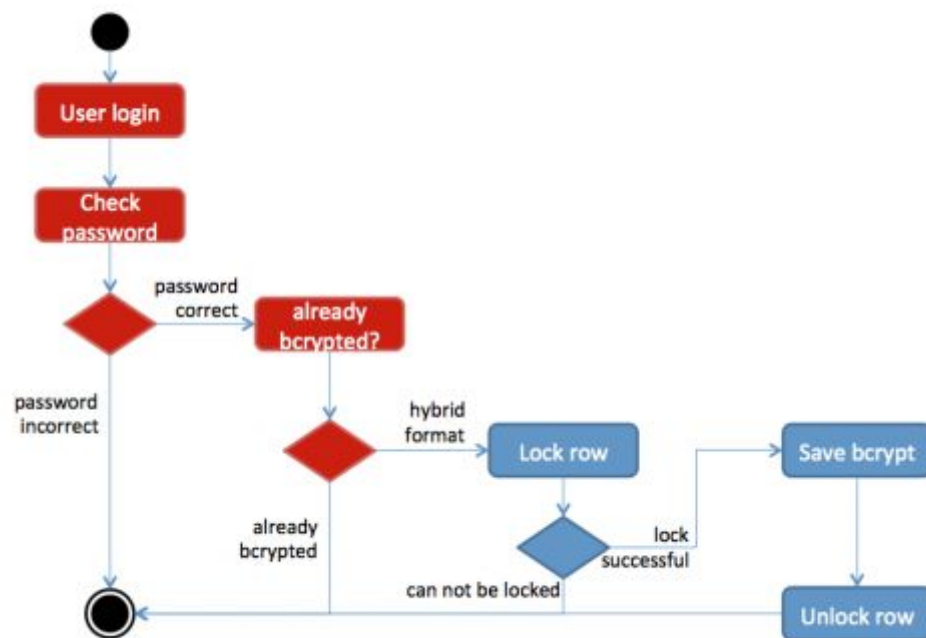


IMAGE14. Passwords converted into “pure” BCrypt format(Dr. Christian Winkler, 2013)

In general, BCrypt encryption is cooperated with cost parameters; and with the additional generation of random salt to the password encryption, it will directly connect into a group of 60 words on the back of a salt yuan Hash when completed.

## References:

IMAGE13, Dr. Christian Winkler, (2013), [image] Available at:

<http://blog.mgm-tp.com/2013/02/securing-your-password-data-base-using-bcrypt/> [Accessed 5 Dec. 2018]

IMAGE14, Dr. Christian Winkler, (2013), [image] Available at:

<http://blog.mgm-tp.com/2013/02/securing-your-password-data-base-using-bcrypt/> [Accessed 5 Dec. 2018]