Secured Entrance System for Facilities

Final Project Report

Sahil Gupta

Avnish Kumar

ECE 4894 Embedded Computing Systems

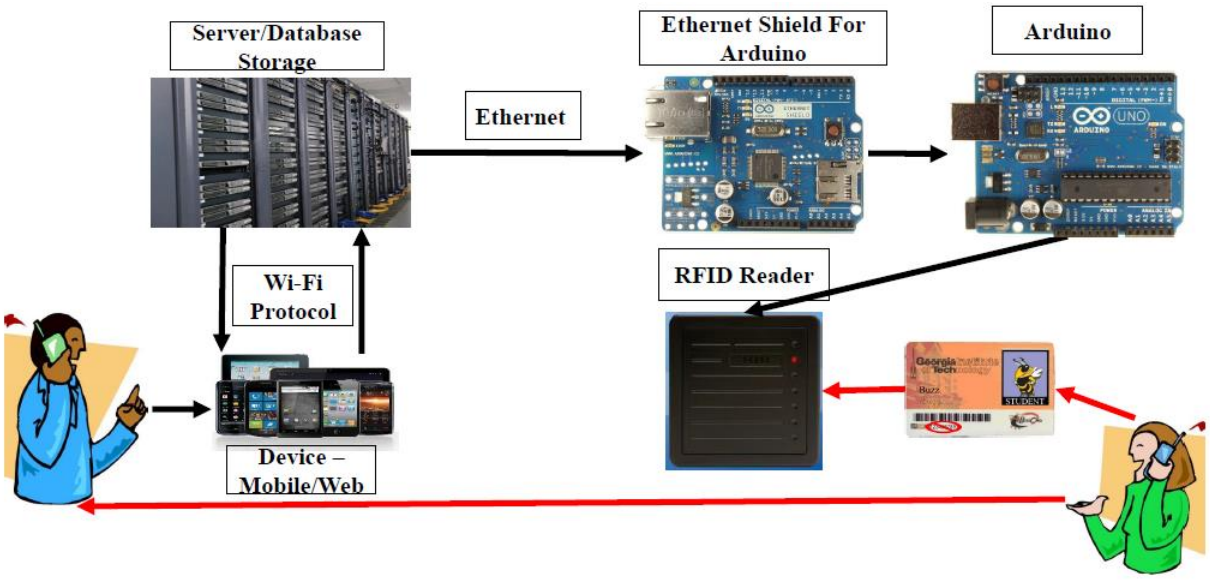
School of Electrical and Computer Engineering

Date Submitted

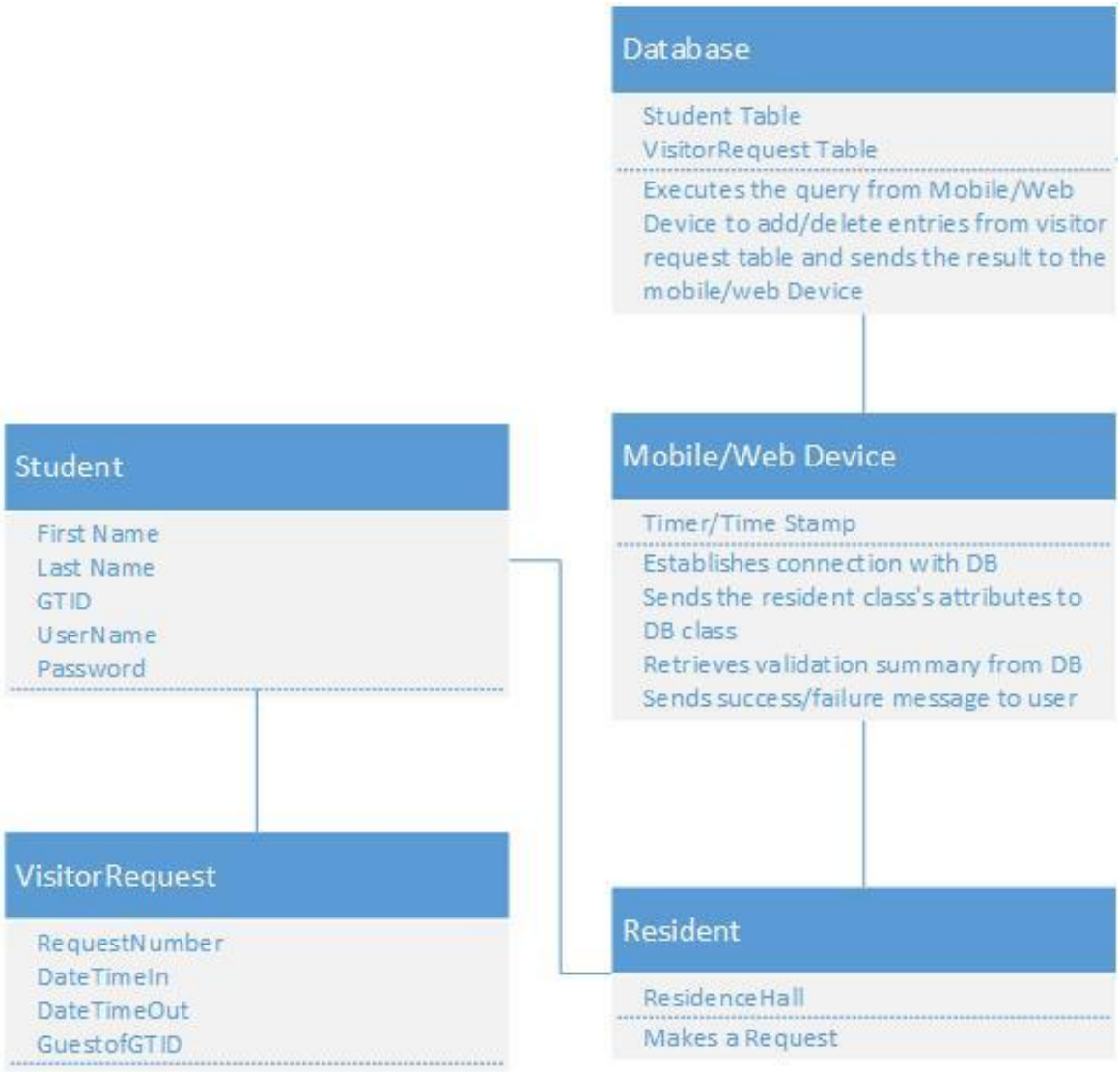
12/03/2013

The secured entrance system for facilities is a consumer grade door lock security system under which the person residing in the facility can authorize a visitor to enter the facility by putting in a request in the database. The resident can use a java based application to enter the credentials of a visitor in a MySQL database. Once a visitor has been entered in the database, that visitor is allowed to enter the building for the following 10 minutes using an RFID tag. For our project, we simulated the infrastructure of Georgia Institute of Technology. The Java application is used to enter the visitor's name and GT ID. The visitor can then use his/her GTID on a RFID reader, which is connected to Arduino UNO. Arduino UNO parses the card to a tag, and checks the database to see if any GTID in the visitor table has a matching tag. If it has a matching tag, the Arduino UNO sends a signal to specify that the user can enter provided time limit is satisfied.

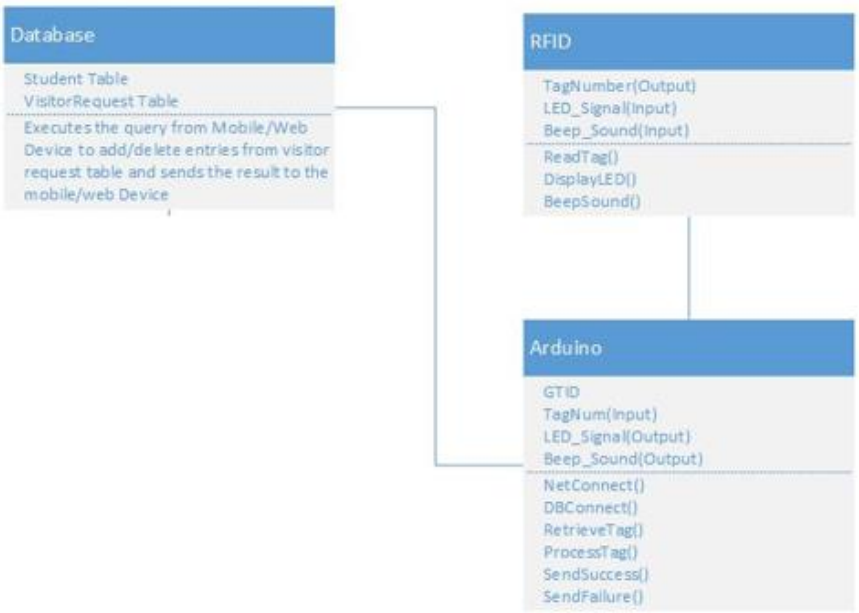
For the entire project, Arduino UNO microcontroller was used. An Ethernet shield was also connected to UNO to help UNO access the database. An RFID reader, which uses the Wiegand standard was also connected to the digital pins of UNO. Overall, all the manufacturing cost of this item is $86 ($51- Arduino UNO + Ethernet shield, and $35 for RFID reader). The following figure shows the overall structure of the project:



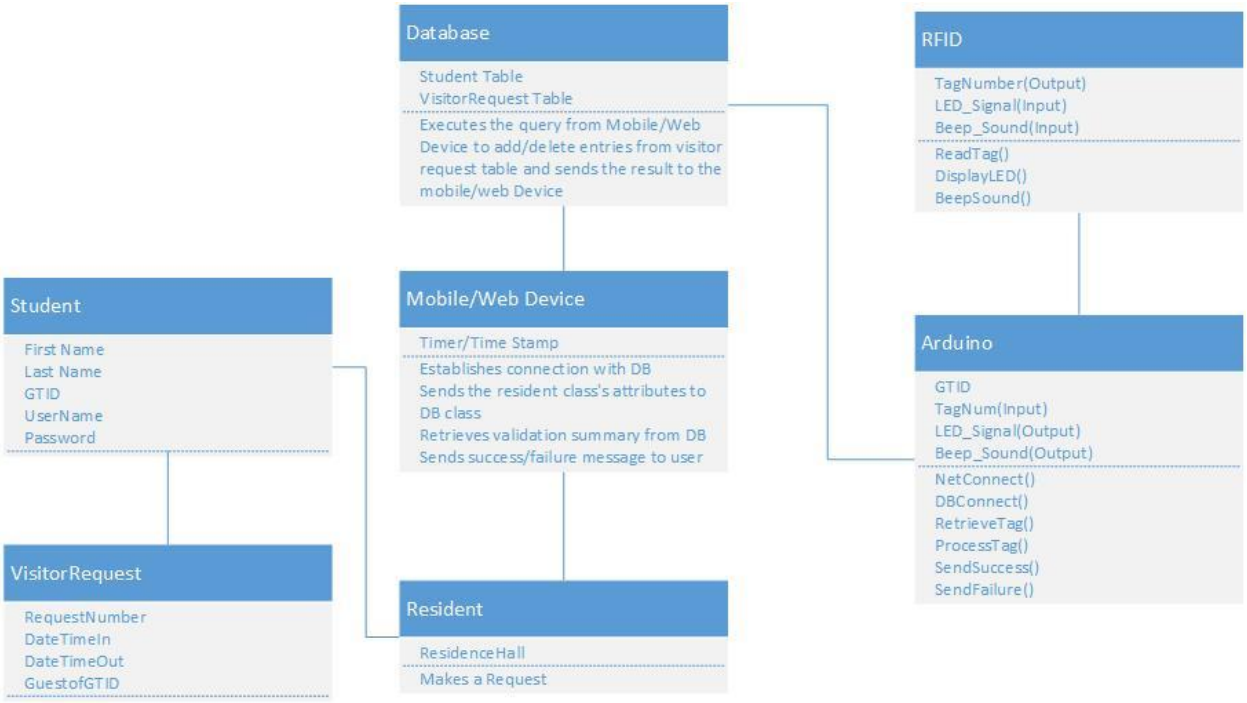
The secured entrance system for facilities has two main modules. The software module contains the database and the web application, whereas the hardware module contains the Arduino, its Ethernet shield and the RFID reader. The class diagrams for the software module is below:



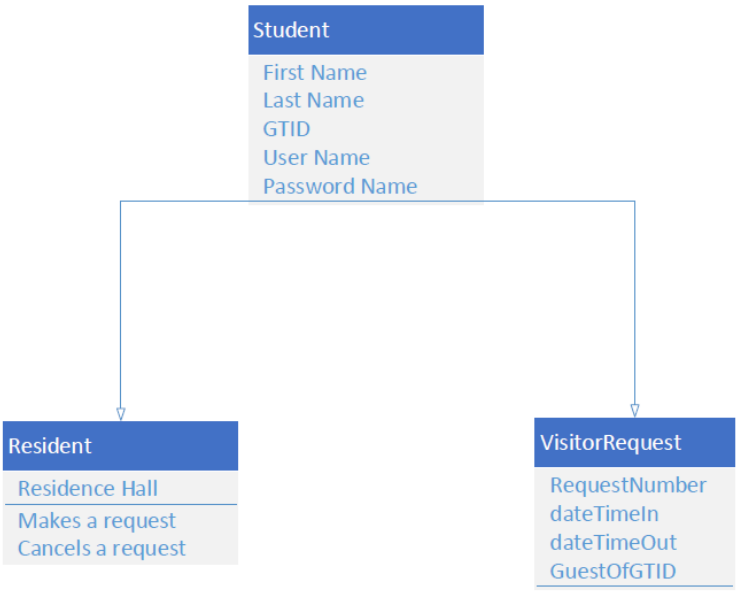
The class diagram for the hardware module is below:



The class diagram for the entire system is below:

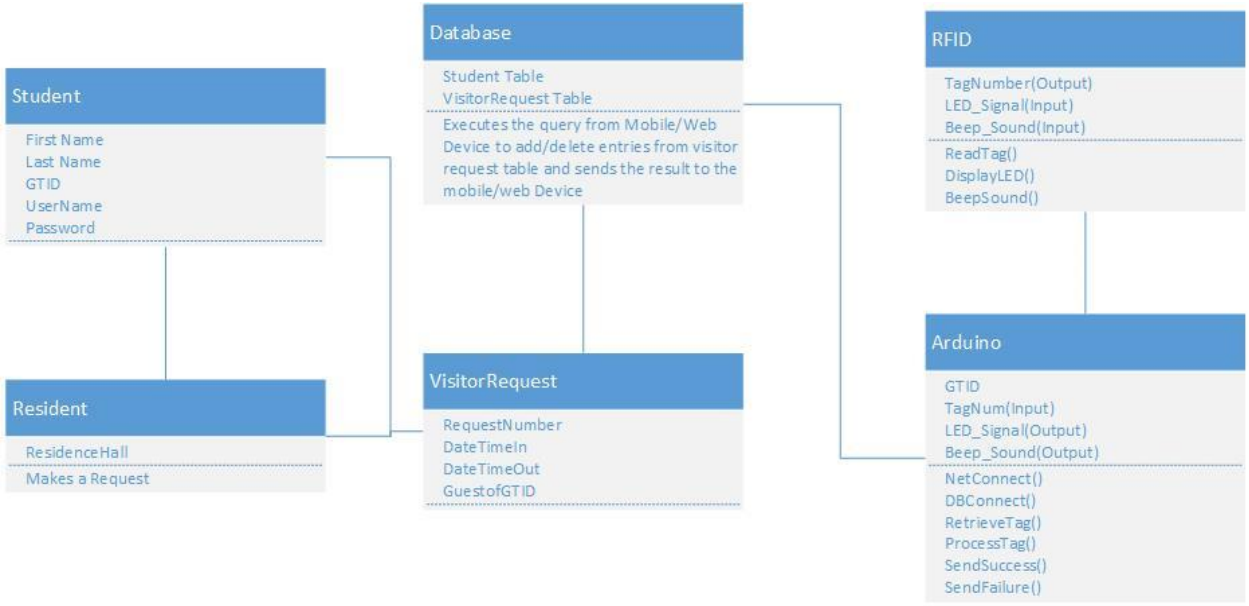


The object diagram for the software module is below:

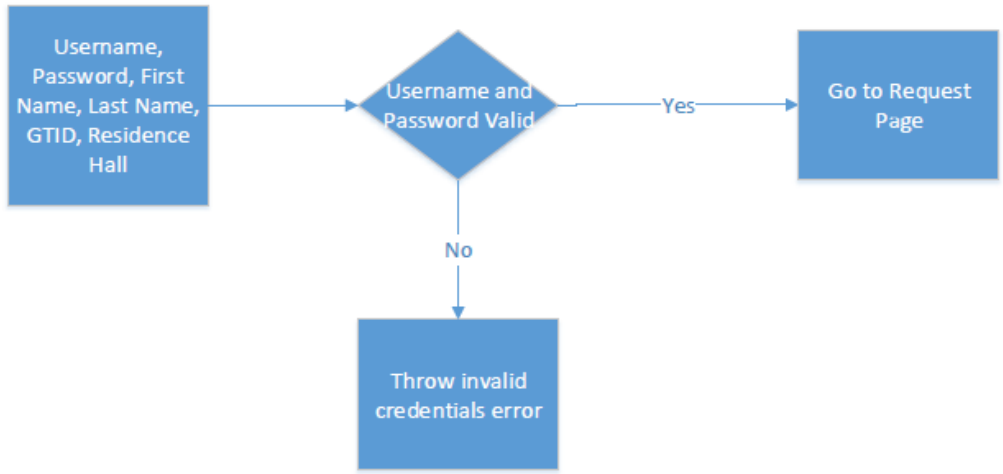


The object diagram for the hardware module is below:

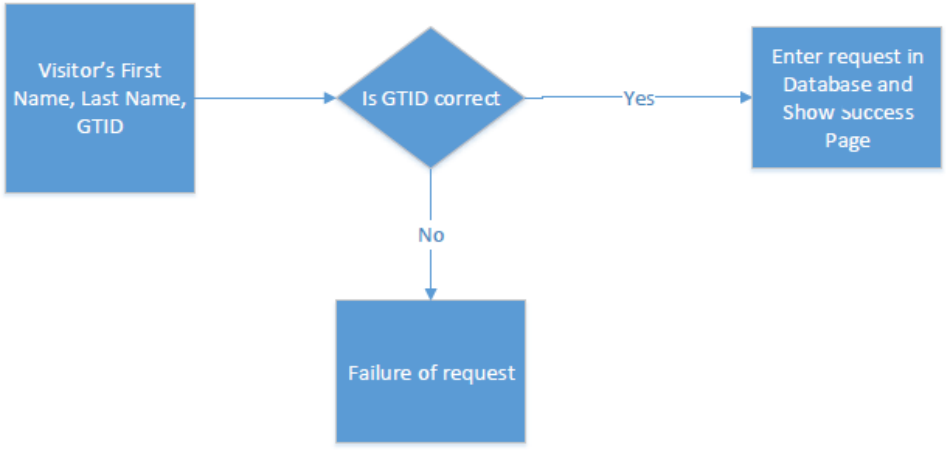
The object diagram for the entire system is below:



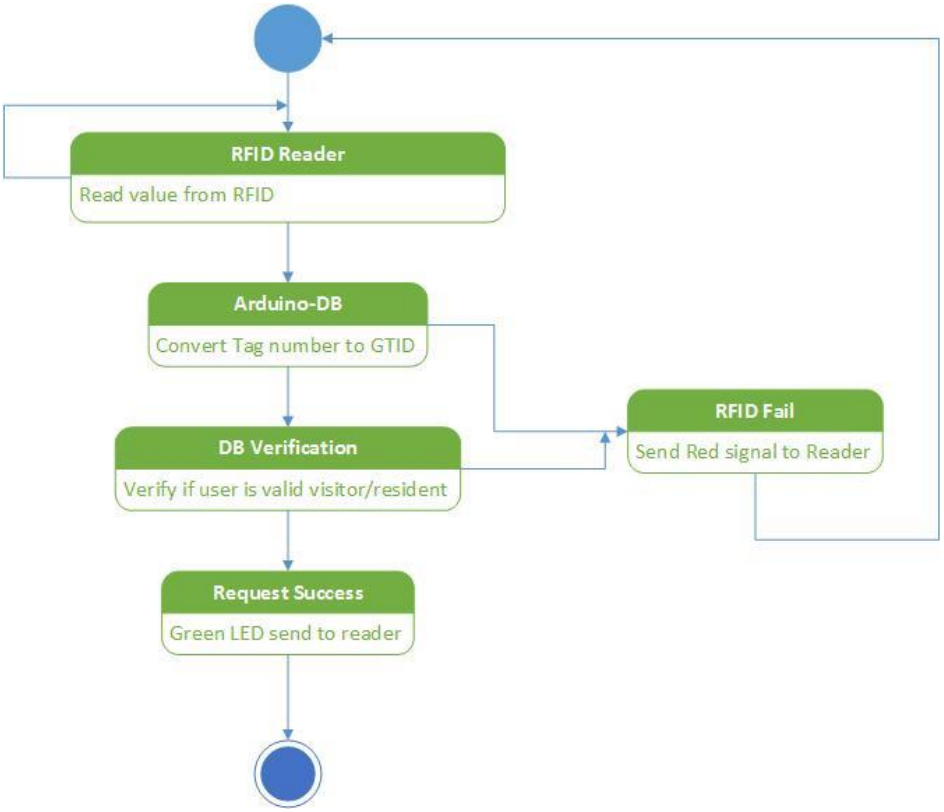
The sequence of events in each module can be represented in the state diagram. The state diagram of the login page is following:



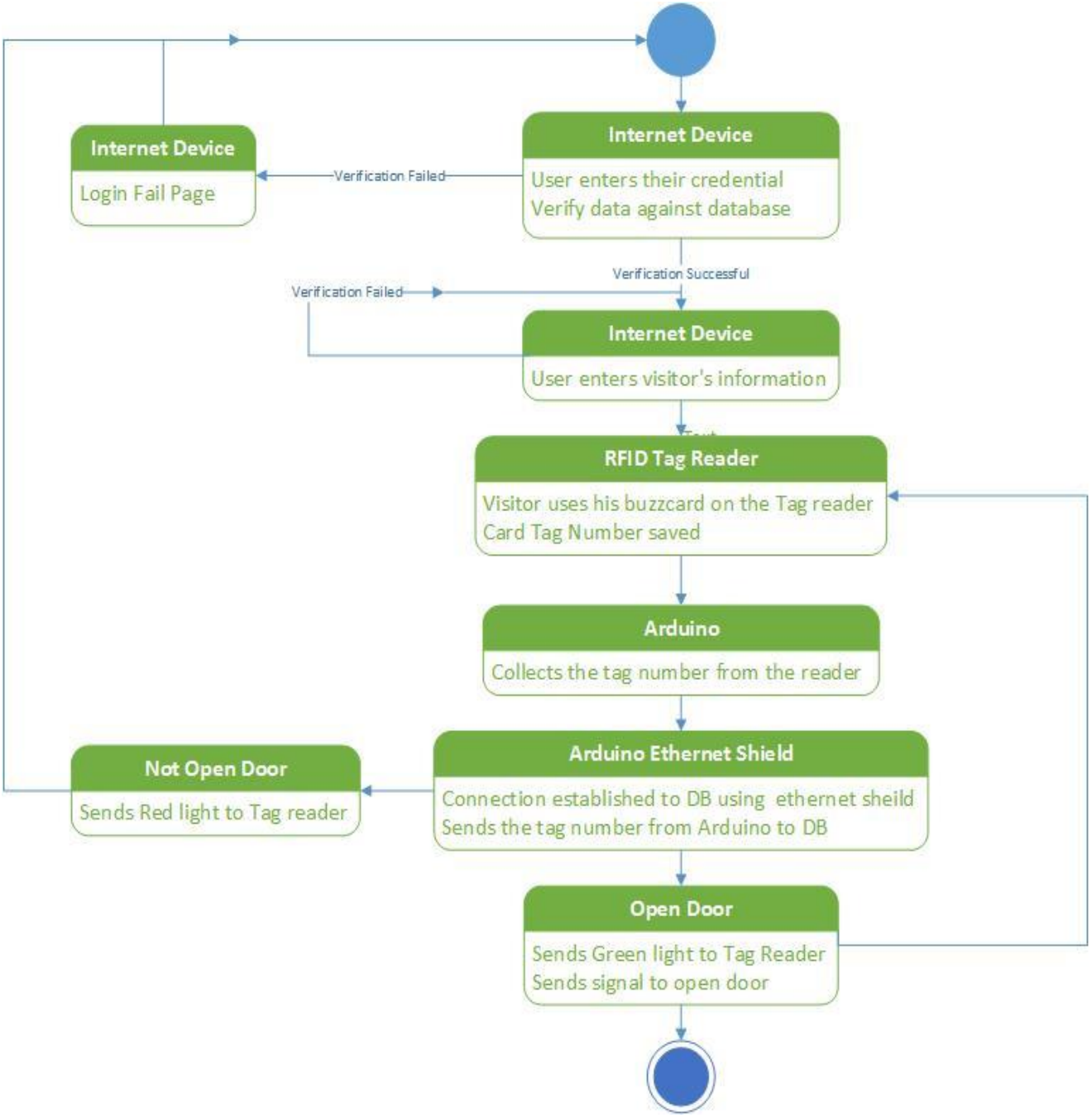
The state diagram for the request page is following:



The sequence of events in the hardware module is following:



The state diagram for the entire system is following:



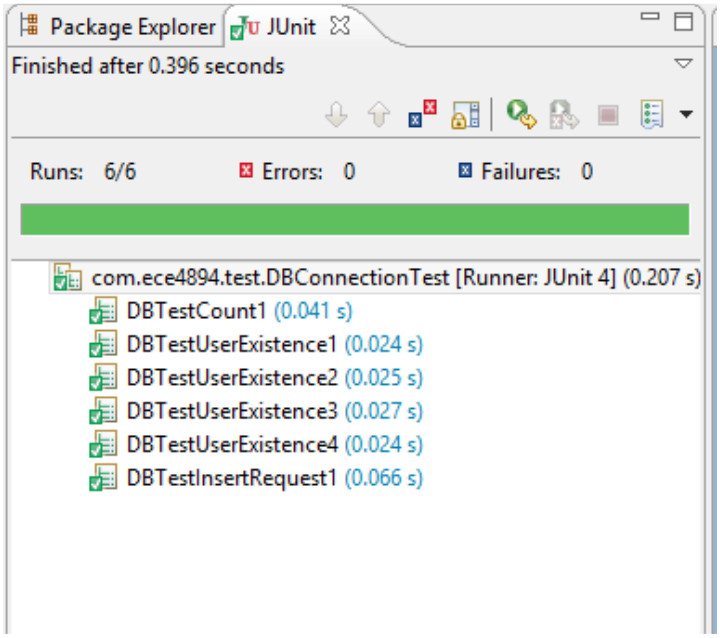
Testing

Both, the hardware module and the software module has been completely implemented and tested. For the software module, the testing performed on the code primarily involved JUnit Testing. The tests are written to check the connection with the database. Further, tests were also written to check if the query to the database insert the request was being made.

Unit Tests:
Tests Applied to check the services such as database connection and correct insertion into the database.

1. @Test//correct access should result in success as expected
DBTestUserExistence1();
2. @Test//correct access should result in success as expected
DBTestUserExistence2();
3. @Test//wrong password should throw an error as expected
DBTestUserExistence3();
4. @Test//wrong username should throw an error as expected
DBTestUserExistence4();
5. @Test//put request for a new visitor
DBTestInsertRequest1();
6. @Test//check that the count of tuples matches as expected
DBTestCount1();

Results from the unit tests:



The hardware module has four major components: connecting the Arduino to the RFID reader, parsing the number from the RFID reader, connecting to the database using the shield and then sending queries to the database. All of the four components can be individually tested. Initially, the RFID reader's connection was tested by writing a program that prints the binary of the inputs. The RFID reader was consistently printing binary, hence showing that the RFID had a stable connection with the Arduino.

For the second component, the Wiegand interface was understood and then the number was parsed from binaries to an unsigned long. It was tested by comparing the number with the tag written on the GTID. Multiple different RFID tags were tested and they all came out successful.

To test the third component of the hardware module, a confirmation message was printed on the screen to tell whether the connection to database was successful or not. Finally, for the fourth component, values were sent to the database through Arduino. These values were then monitored in the database, to see if they were actually entered. They were tested by checking if the correct values are being entered in the correct column.

Once both the hardware as well as the software component had been tested, the entire system was tested by doing black box testing multiple times with different GTIDs.

Software Module Level Test Plan:

Black box testing:

Inputs to Login Page	**Outputs**	**Example Input**
Empty values for any of the username, password, first name, last name, GT ID and residence hall.	Information not entered error	Not Applicable
Username is not a valid alpha-numeric input	Invalid username value type	Fhsdfu12!@#

GTID is not a 9 digit number	Invalid GTID value type error	Jbrrejkbt432
First name is not a proper alphabetical string	Invalid first name value type	SAHIL4324&%^
Last name is not a proper alphabetical string	Invalid last name value type	GUP34235!
Residence hall is not the three-letter code for the hall.	Invalid residence hall code value type.	DSHJA

Inputs to the Request Page	**Outputs**	**Example Input**
Empty values for any of the First Name, Last Name and GTID	Information not entered error	Not applicable
First name is not a proper alphabetical string	Invalid first name value type	SAHIL4324&%^
Last name is not a proper alphabetical string	Invalid last name value type	GUP34235!
GTID is not a 9 digit number	Invalid GTID value type error	Jbrrejkbt432

Clear Box testing:

Tests to check several paths/conditions in the loops:

Inputs to Login Page	**Outputs**	**Example Input**
Invalid username and password value	Invalid credentials error	Username -Ksmith4 Password – abcd123 When the user Kelly Smith does not exist as a member in Student table in the database
Wrong GTID value entered	Invalid GTID error	9043567276 when the GTID does not exist in DB
Wrong residence hall value entered	Invalid residence hall error	NSA when the hall should be NAS
Correct username, password, first name, last name, GT ID and residence hall values.	Request Page Shown	Not applicable

Inputs to Request Page	Outputs	Example Input
Invalid first name, last name or GT ID value	Invalid information for visitor	First Name – Kelly Last Name – Smith GTID - 4378943242 When the user does not exist with such a name.
Valid first name, last name and GT ID	Success Page Shown/Database updates the entry	Not applicable

Def-use test cases:

For Login Page:

Variable Definition	Variable Assignment
public Resident CurrentUser;	CurrentUser.setUserName(userNameTextField.getText()); CurrentUser.setPassword(passwordTextField.getText()); CurrentUser.setFirstName(FirstNameTextField.getText()); CurrentUser.setLastName(LastNameTextField.getText()); CurrentUser.setResidenceHall(ResidenceHallTextField.getText()); CurrentUser.setGTID(GTIDNumber);
Test if the user entered everything correctly for his/her/resident's information	If the user entered correctly, then the assignment to CurrentUser/Resident should be correct as well. For instance, in login page attempt getting in with user Sahil Gupta with username: sgupta23 and password: abc8; residence hall= NAN; GT ID = 902648052.

For Request Page:

Variable Definition	Variable Assignment

VisitorRequest request;	request.setGuestOfGTID(currentUser.getGTID());
	request.setDateTimeIn(new BigDecimal(new Long(System.currentTimeMillis()).toString()));
	request.setDateTimeOut(new BigDecimal(new Long(System.currentTimeMillis() + 600000).toString()));

Test if the user entered everything correctly for the visitor's information	If the user entered correctly, then the assignment to request/visitor should be correct as well. For instance, in request page create a request for first name and last name Eric Cart respectively, and GTID 902348523. This assignment should happen correctly.

Hardware Module Test Plan

Black Box Testing Plan:

The Arduino module has only one input: tag number from the RFID (Student ID). When someone uses their student ID on the RFID reader, the reader translates the RFID tag to a unique number. For black box testing, we used different types of RFID to see if we get the result we should be getting. The output in our case is different signals/lights we see on the RFID reader. A red light indicates that the credential (the RFID tag) wasn't correct and that the door can't be opened. A green light indicates that the credential (the RFID tag) was correct and that the visitor/resident can enter the building.

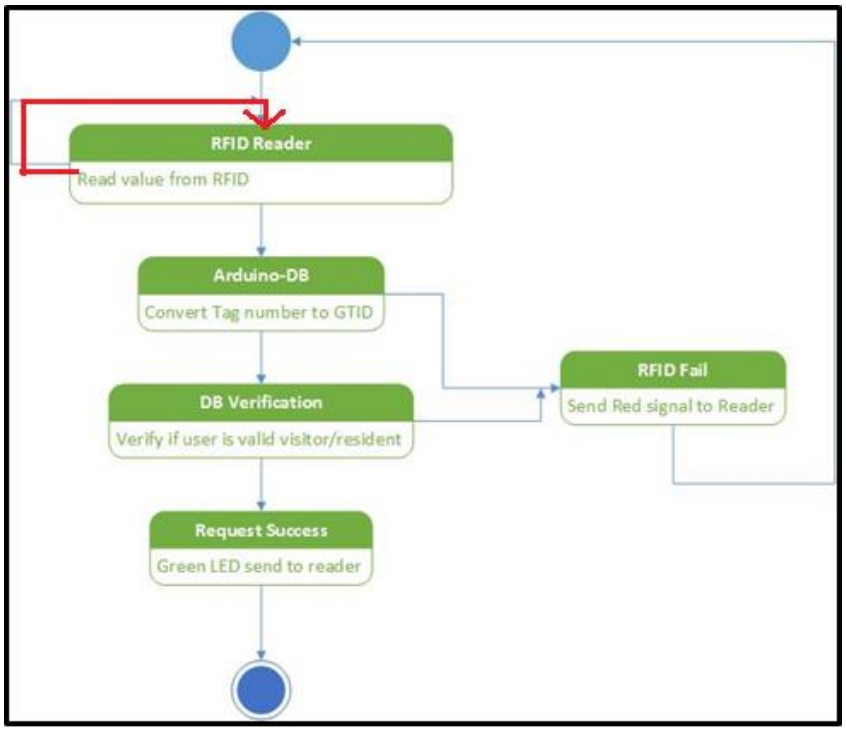
Based on this, following are the black box tests that can be performed in the Arduino module:

a. Correct type of RFID tag (GT ID versus UGA (or some other kind) ID) – This test will see if only the GTID is being interpreted and not any other kind of ID. This can be tested by using different types of ID's and seeing if the door signal turns green or red.
b. Test for Resident Tag – This black box test will test whether the door opens up for a resident. The input in this case will be a valid ID of the resident of the building, and the output should be that the door open signals should be true.
c. Test for Non-Resident Tag – This black box will test whether the door opens up for a GTID who doesn't belong to that residence hall. The input will be a valid ID of a non-resident of that building, and the output should be that the door open signals should be false.
d. Test for Active Visitor Tag – This black box test will test whether the door opens up for a GTID of a student who is in the active visitor table for that residence hall. An active visitor is a visitor who was added in the last 10 minutes (the time set during specification for the visitor to enter the building). The output should be that the door open signals should be true.
e. Test for In-Active Visitor Tag – This black box test will test whether the door opens up for a GTID of a student who is not an active visitor for that residence hall anymore. This student would have been an active visitor earlier and his access timed out (> 10 minutes). The output should be that the door open signals should be false.

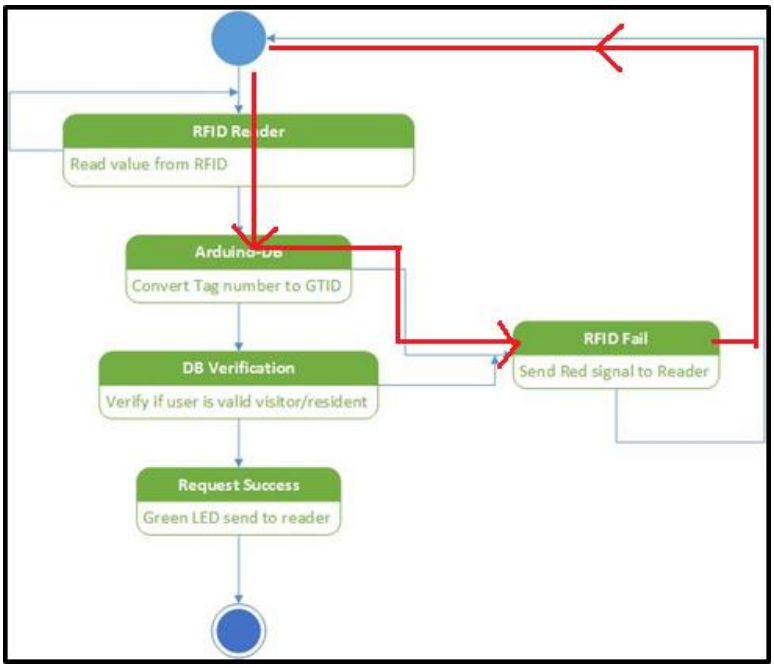
Clear Box Testing Plan

Following tests were performed to cover each path of the algorithm:

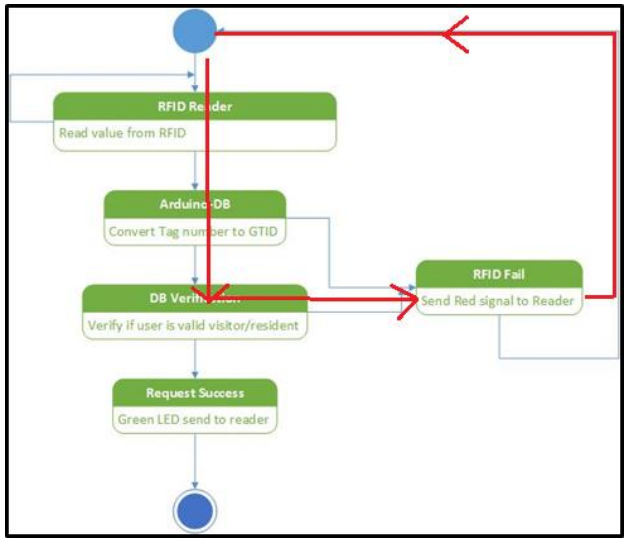
a) Check if no RFID tag is near the reader, if it is passing any values to the Arduino. If there is no tag near the reader, the reader should not pass any value to the Arduino. This can be tested by not having any tag near the reader and seeing if any value is passed to Ardunio. This will test the following path in the state machine:



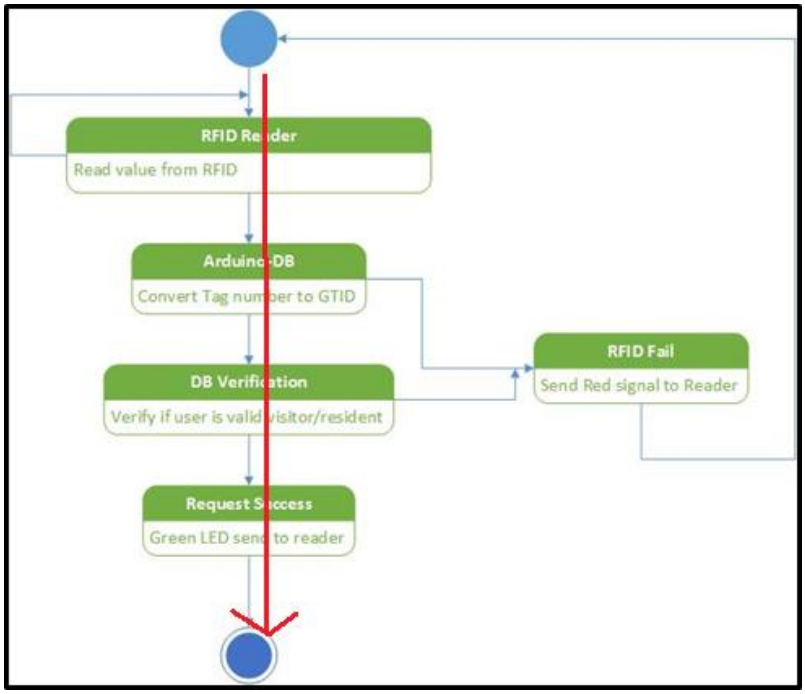
b) Check if Arduino sends a non-Georgia Tech ID for processing to see if it is a visitor or resident. This can be tested by using a non-Georgia Tech ID and seeing the output. The output should be a red LED. Following branch is being tested in this case:



c) Check if a non-resident Georgia Tech or an in-active visitor is let inside the residence. This can be tested by using a non-resident tag and making sure the Red LED turns on. This tests the following branch:



d) Check if a resident or a active visitor is let inside the building. This can be checked by using GTID of a resident or an active visitor. The green LED should turn on. This test the following branch:



Summary

A total of roughly 150 man hours were spent by the team of two for this project. Major changes made since the last design cycle primarily involve the hardware module and database hosting. Currently the database is being hosted on the local host due to networking issues we faced with connecting the Arduino to the server hosted online on the internet. Hence, we had to change our design strategy from client side processing on the Arduino UNO device to server side processing on the local host server. In terms of the hardware module, Ethernet shield is not being used for its intended purpose of connecting to the internet. Instead, we are using it to connect to the local host which can also be done with some further work with the Arduino UNO module as well.

Technologically, we learned a lot about microcontrollers and how they work. We learned starting from the basics of microcontrollers such as controlling a LED to a more advanced step of connecting it to a database. From the software side, we had to learn Python, which was used for

server-side programming. We also honed our Java skills by making the front end application for users. In attempts to connect to database, we also learned a lot about networks and what different IP addresses mean. We also learned about Wiegand interface and how the Wiegand interface is interpreted.

Besides the technical knowledge we gained, we also learned how to properly plan out a large project and the different steps involved in carrying out a project, starting from planning to implementation and testing. We also learned about the various UML standards and how they can be used to represent information.