
Bilkent University

Department of Computer Engineering

# Senior Design Project

*Automated Attendance Taking System (AATS)*

## Analysis Report

Alba Mustafaj, Argert Boja, Ndriçim Rrapi, Rubin Daija

Supervisor: Selim Aksoy
Jury Members: Ibrahim Korpeoglu and Hamdi Dibeklioglu

November 9, 2018

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Contents

# Analysis Report

*Automated Attendance Taking System (AATS)*

## 1. **Introduction**

The best way to understand a topic is by being involved in it. This is one of the main reasons why in universities participation in lectures is highly advised and sometimes even mandatory. However, occasionally student tend to skip their lectures but they still want to be counted as present, and for this they use several ways. This fact is widely known even by the professors, nevertheless, it is sometimes impossible to avoid it. Despite the alternative ways that can be used for checking whether all signatures are one-to-one, this can lead to inefficient usage of lecture time.

As we all might have experienced, the traditional way of checking for attendance is by passing a paper which contains all student names so that they can sign for themselves. This method of taking attendance is yet another reason for irregularities in this system. This is because sometimes the professors might forget to bring the attendance paper, or the students might forget to return the attendance paper. Furthermore, some students might come a few minutes late and they forget to sign the attendance.

This project aims to improve the methods of attendance taking, which will lead to less responsibilities for instructors and more accuracy for students. By using face recognition, we will be able to identify all the students present in class. Furthermore, by conducting several checks through the class hour there will be no possibility of cheating. Both students and instructors will be able to check the results on an application that will be build. Even if there is any inaccuracy, the students will be able to still be counted as present by letting their professor know that they were present in the taken picture. In this way, the new data will be collected and the system will be trained again in order to adapt to individuals changing their look.

The following sections of this report, will provide a thorough description of our project. Current systems will be described. This will be followed by all the requirements that have been identified by this phase. In addition, system models such as scenarios, use cases, object models, activity diagrams and activity sequences will be provided, to better understand the project. It is significant to note that a part of our design is based

on the fact that this system would, if deployed, be in connection with the current university system, as such having access to different information within the university's database.

## 2. Current System

A part of the research done prior to this report was related to existing solutions of the problem of attendance taking. The results of our research show that there are actually a lot of applications which deal with attendance taking and a small number of them are shown below.

- AccuClass is a web-based classroom attendance tracking system. AccuClass offers several options for recording the attendance data including using a computer or a laptop, using a time clock or using an Apple iPad, iPod touch, or iPhone. The users have a list of the current students and the attendance taker has to mark the students one by one.[1]
- MyAttendanceTracker is another web-based application which is used to help teachers track the attendance of their students. The admins/teachers manage the list of the students, courses and related reports. They are responsible for marking each student one by one. [2]

These were two of the existing products that are already available on the market. As described they consist mainly in giving users the opportunity to manage the attendance manually by offering applications that are easy to use and online.

However, we could not find any existing product that would be handling the process of attendance taking by making use of facial recognition. There were a lot of research papers and senior projects related to the issue, yet none of them had been concluded into an actual product.

The solution that we propose aims to combine both the old idea of an attendance tracker application and facial recognition into one single product. The users of this product should have little or no manual work to do as the process will be fully automated.

## 3.Proposed System

### 3.1 Overview

Automated Attendance Taking System (AATS) aims to automate the whole process of taking attendance in class. This system will be aided cameras that will be used to take in-class attendance. The cameras will be frequently taking pictures between certain

periods to ensure that the students who are attending the class are marked present. These pictures are then analyzed by using image recognition to identify the students that were present during the lecture. This information is then automatically retrieved and sent to the university's database, where the attendance data is instantly updated. The process eliminates all the hassle that the students have to go through while signing the attendance sheet but it also aids professors who might occasionally forget to deliver the attendance paper. It also eliminates the time wasted by professors or their assistants to manually upload the attendance data online.

### 3.2 Functional Requirements

- The user should always be informed when the picture is being taken by a notification.
- The user should be able to make an objection in case he was present in the class but the system marked her/him absent.
- In case of an objection, the professor as a higher authority should be able to accept or deny the objection.
- The professor should be able to view attendance statistics for his/her classes.

### 3.3 Non-functional Requirements

- Performance: The system should be able to process at least a picture from each class in the University within the timeframe of a lecture hour.
- Capacity: The system should be able to take a picture of the whole class.
- Confidentiality: A picture cannot be taken without firstly notifying the user.
- Accessibility: The application will use university credentials for a one time log in and is accessible by all students.
- Availability: The system shall be available to the users at any time of the day.
- Reliability: In case of a server issue and inability to process the data, the images will be stored in the database for later processing.

- Security: The communication within the systems will be encrypted. Only authorized individuals can access will have access to the data.
- Accuracy: The system should be able to identify multiple individuals with a high degree of similarity of their facial features.

## 3.4 System Models

### 3.4.1 Scenarios

#### 3.4.1.1 Scenario 1

| | |
|---|---|
| *Use case name* | StudentNotRecognized |
| *Participating actor instances* | Albert:Professor<br>John:Student |
| *Flow of events* | 1. It is the end of the class and the AATS is done checking who is in class. John was asleep during most of the class and thus receives the notification on his phone that he was not counted as present during this hour.<br>2. John believes that he should be counted as he was in class, thus he objects the result in the AATS mobile application. Then he proceeds on going to Albert to argue his objection.<br>3. Albert is notified of the objection by the AATS mobile application. He opens the application and investigates the pictures taken by the system. He sees that John was asleep during the time when the pictures were taken.<br>4. John comes to argue his case, however the professor does not accept his case due to him technically not being present during the class. |

#### 3.4.1.2 Scenario 2

| | |
|---|---|
| *Use case name* | pictureTakingNotification |
| *Participating actor instances* | Albert:Professor<br>John:Student |
| *Flow of events* | 1. It is the time when the system is scheduled to take the picture of the class for attendance.<br>2. The system sends a notification to John as well as Albert so that they know the picture is being taken. |

#### 3.4.1.3 Scenario 3

| | |
|---|---|
| *Use case name* | resultsAreIn |
| *Participating actor instances* | Albert:Professor<br>John:Student |
| *Flow of events* | 1. It is the end of the class and the system is done checking attendance. Thus it send a notification to John and Albert.<br>2. Albert sees the notification and is able to learn how many students were present and if there is an objection.<br>3. John receives the notification and is able to see if he was counted in and, if he wasn't, he has the possibility to make an objection. |

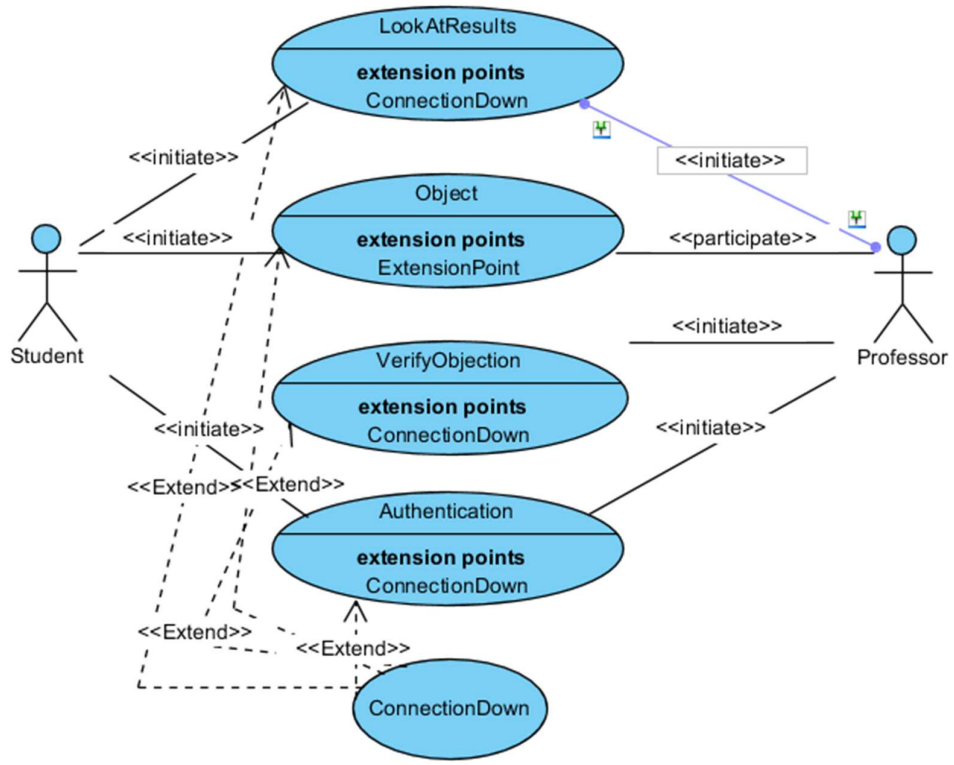
#### 3.4.1.4 Scenario 4

| | |
|---|---|
| *Use case name* | StudentAuthentication |
| *Participating actor instances* | John:Student |
| *Flow of events* | 1. John opens the AATS mobile application. He selects to log in as a student and inputs his credentials.<br>2. The system checks his credentials against the ones found in the database.<br>3. The credentials are correct and as such John is shown his class attending results and statistics. |

#### 3.4.1.5 Scenario 5

| | |
|---|---|
| *Use case name* | ProfessorAuthentication |
| *Participating actor instances* | Albert:Professor |
| *Flow of events* | 1. Albert opens the AATS mobile application. He selects to log in as a professor and inputs his credentials.<br>2. The system checks his credentials against the ones found in the database.<br>3. The credentials are correct and as such Albert can see different statistics about the attendance in his classes. |

### 3.4.2 Use Case Model




#### 3.4.2.1 Use Case 1

| | |
|---|---|
| *Use case name* | Object |
| *Participating actors* | Initiated by Student<br>Communicates with Professor |
| *Flow of events* | 1. The Student objects the fact that he/she was not counted in, in the AATS mobile application.<br>2. AATS responds by sending the objection to the Professor.<br>3. The student receives a notification on the Professor decision |
| *Entry condition* | • The Student is logged into the AATS mobile application. |
| *Exit conditions* | • The Student receives a notification about the Professors decision. |
| *Quality requirements* | • The Student's objection should not be sent later than 10 seconds after it filing to the professor. |

#### 3.4.2.2 Use case 2

| | |
|---|---|
| *Use case name* | SeeResults |
| *Participating actors* | Initiated by Student or Professor |
| *Flow of events* | 1. The Professor or the Student requests to see the results regarding the attendance.<br>2. The system receives the request and presents the adequate results to the party. |
| *Entry condition* | • The Student or the Professor is logged into the AATS mobile application. |
| *Exit conditions* | • The Student or the Professor exits the application. |
| *Quality requirements* | • The results should be presented within 3 seconds of the request. |

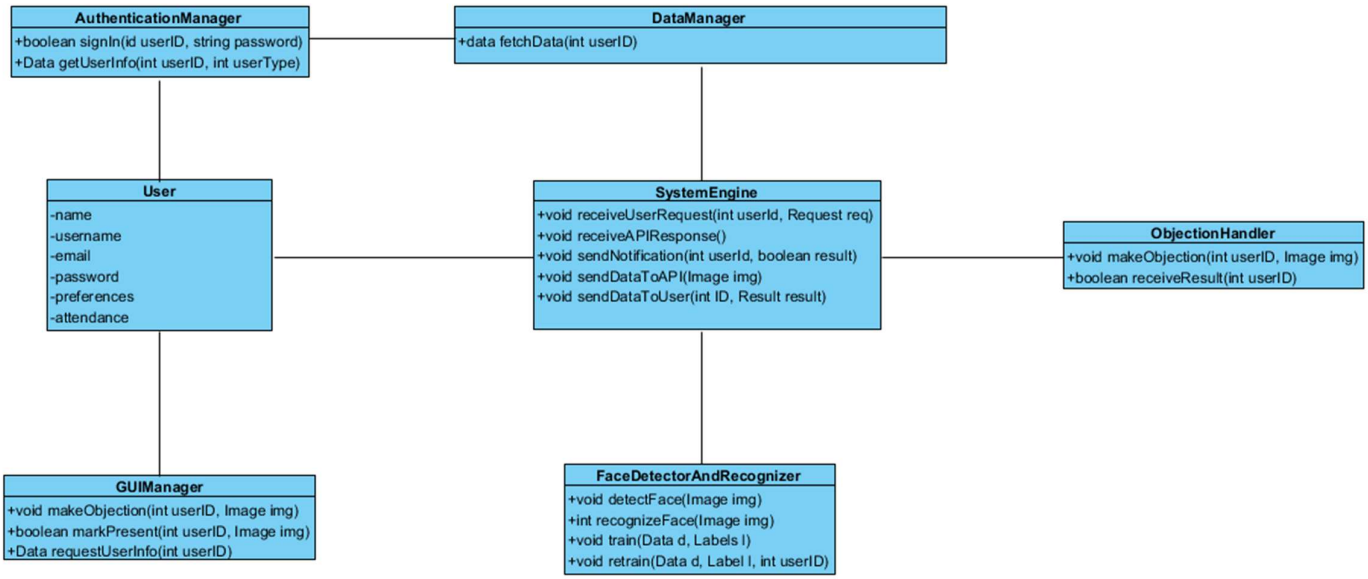
#### 3.4.2.3 Use case 3

| | |
|---|---|
| *Use case name* | Authentication |
| *Participating actors* | Initiated by Student or Professor |
| *Flow of events* | 1. The Professor or the Student enter their credentials on the AATS mobile application.<br>2. The application sends a request to the system to check the credentials.<br>3. Provided that the credentials are correct the appropriate welcome screen is displayed. |
| *Entry condition* | • The Student or the Professor opens the AATS mobile application. |
| *Exit conditions* | • The Student or the Professor receive a result on whether they can or not log in to the system. |
| *Quality requirements* | • The results should be presented within 3 seconds of the request. |

### 3.4.2.4 Use Case 4

| | |
|---|---|
| *Use case name* | VerifyObjection |
| *Participating actors* | Initiated Professor |
| *Flow of events* | 1. The Professor receives a notification for an objection.<br>2. The Professor looks at the information provided by the system and makes a decision.<br>3. The Professor submits that decision through the AATS mobile application. |
| *Entry condition* | • The Professor is logged into the AATS mobile application. |
| *Exit conditions* | • The Professor submits his decision. |
| *Quality requirements* | • All parties should be notified within a time span of 10 seconds of all the occurring decisions. |

### 3.4.3 Object and class model




**Authentication Manager:** is used to authenticate a user's information during log in.

**Data Manager:** is used to fetch the data from the database for the different functions of the system.

**User:** holds all the information for the current user.

**GUIManager:** handles all the views of the system, like displaying log in, results etc. .

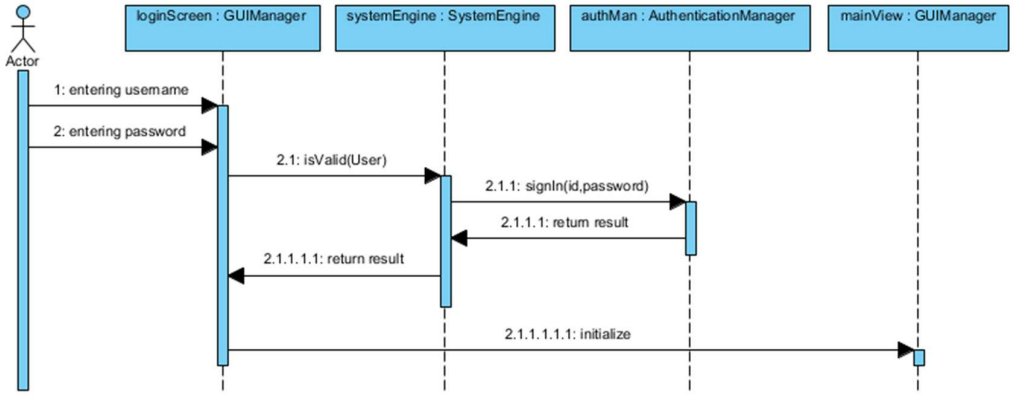
**SystemEngine:** is used to receive user requests and process them. Furthermore it connects all the parts together, so that if a part of the system needs to access the database it has to do it through this.

**FaceDetectorAndRecognizer:** is used to recognize the faces in the system.

**ObjectionHandler:** is used to handle objections that are made by the students, and also the results from the professor's decision.

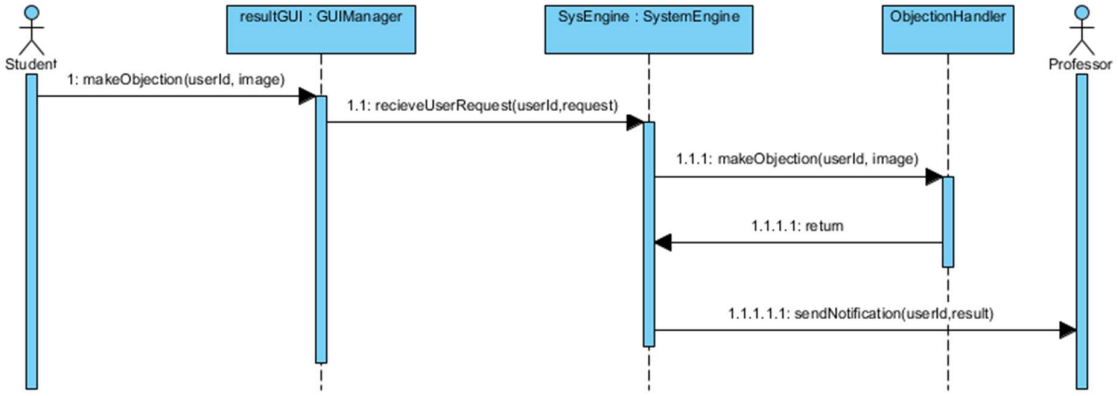
### 3.4.4 Dynamic Models

#### 3.4.4.1 Log in Sequence



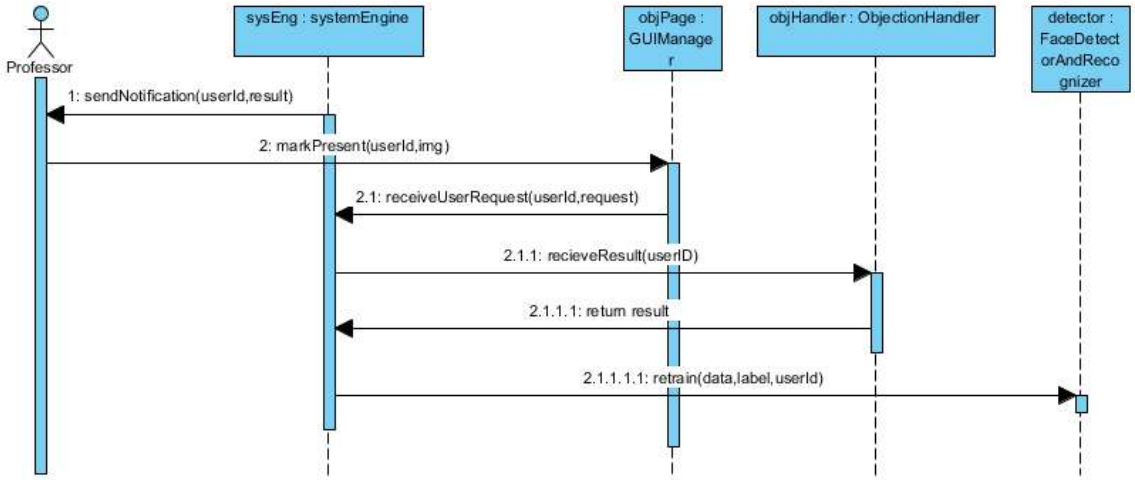

In order to log in a user must enter the password and the username in the log in screen in the AATS mobile application. The GUI manager sends a request to the system to check if the user is valid. The system engine checks in with the authentication manager to check if the credentials are valid. Provided that all the information is valid the GUI manager will open a new main view for the user.

### 3.4.4.2 Make objection sequence



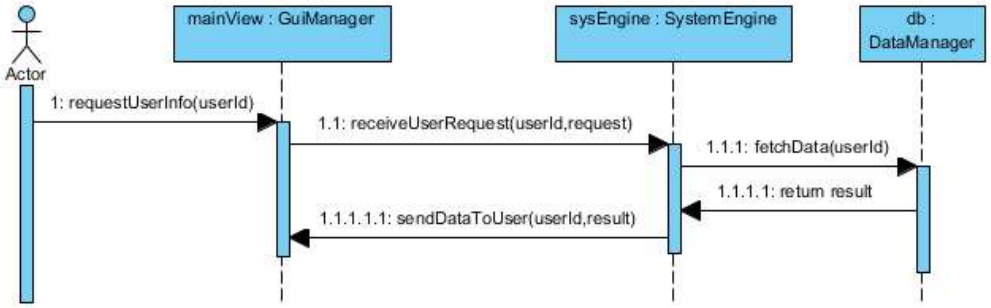

The students initiates an objection by using the application. This request is sent to the system which then forwards this requests to the objection handler. The handler simply accepts this request and then a notification is sent to the professor to notify him of the objection.

### 3.4.4.3 Mark present sequence



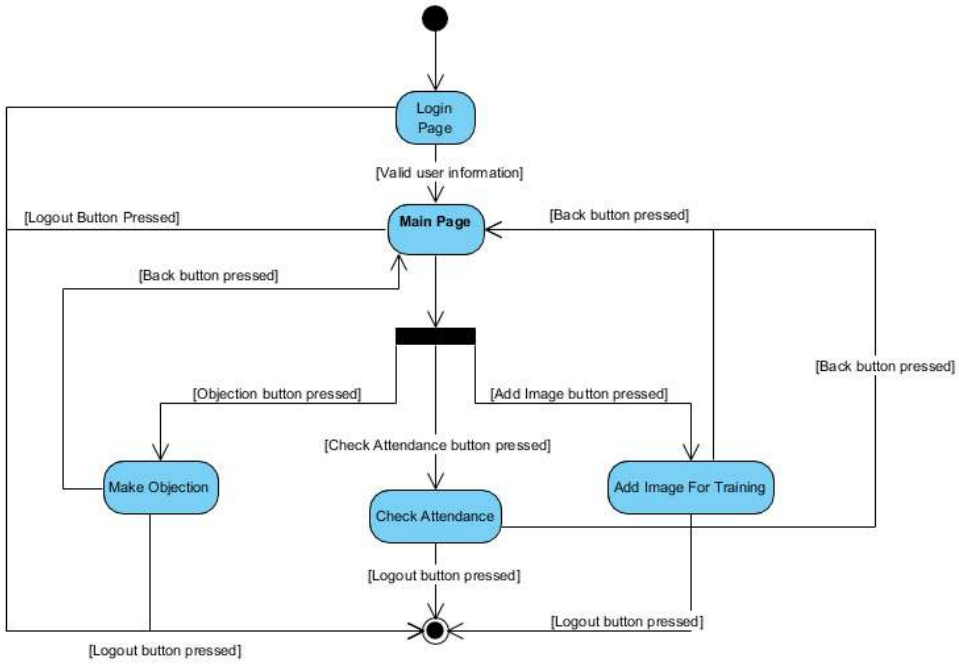

The professor gets a notification from the system about the objection. Then he uses the application to mark the student as present. This result is registered by the objection handler and it is returned to the system engine which trains the face detector for the new unrecognized face.

### 3.4.4.4 Show results sequence

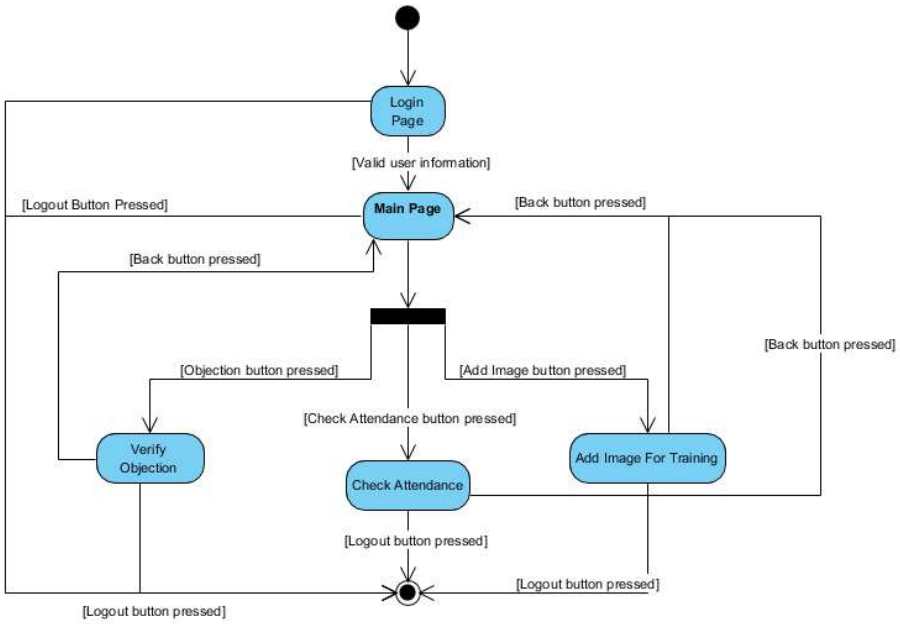



The professor or the student use the application to request the information about their account. The application sends a request to the system which in turn send the request to fetch data to the data manager. The result then is returned to the application which displays it to the user.

### 3.4.4.5 Student Activity Diagram



### 3.4.4.6 Professor Activity Diagram



### 3.4.5 User Interface

The following part consists of the user interface of our application with the corresponding details along with the sketches. This interface represents the client side of our application since the major part of the back-end is done on the server.

#### 3.4.5.1 Sign in

The following figure shows the 'sign in' screen of the application. As it can be seen the user chooses their position as professor or student and then completes the rest of the credentials to sign in to the system. There is no sign up since we assume the data is received from the already existing university database.

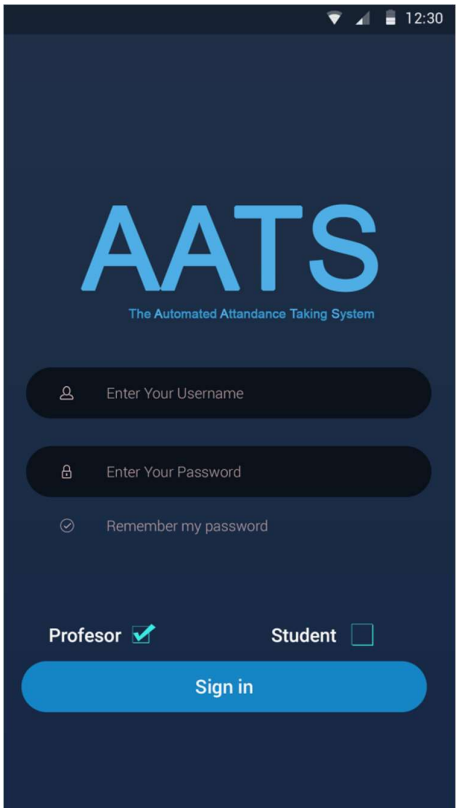



*Figure 1 - Login Screen*

#### 3.4.5.2 User Interface for Professors

The following section consists of the user interface as seen from the professor's point of view.

#### 3.4.5.2.1 Main Screen Professor

The main screen of the professor consists of the personal information of the professor and three buttons. From here the professor can choose to see the list of courses taught, the list of objections made by students and also change the applications settings.

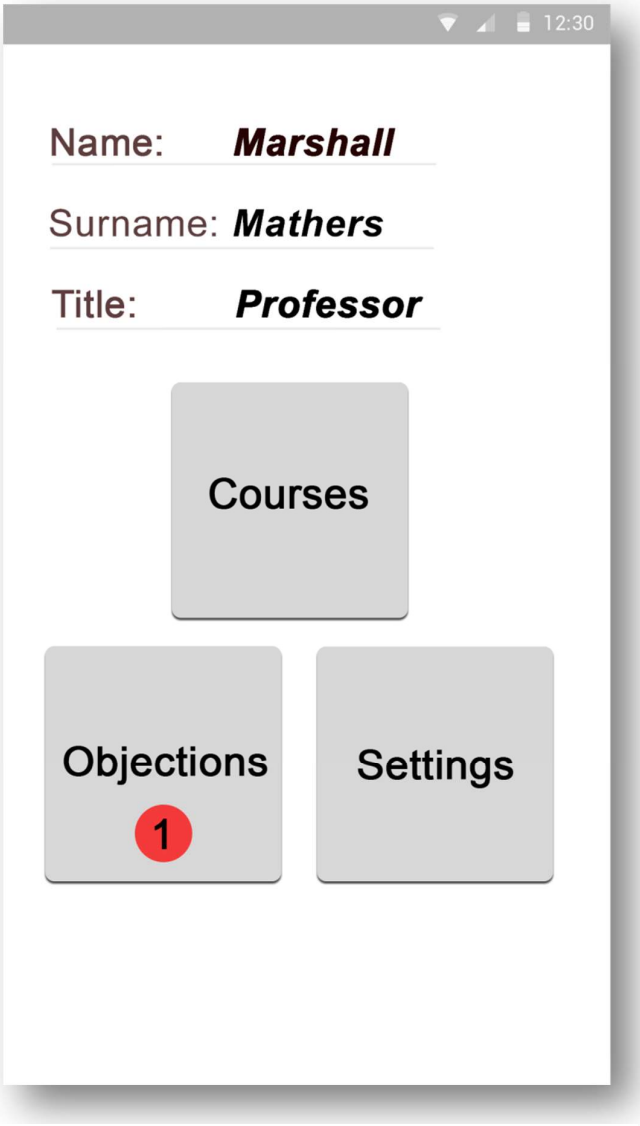



*Figure 2 - Main Screen of the professors*

#### 3.4.5.2.2 List of Courses Screen

The following screen shows the list of the courses taught by the professor and corresponding sections. Each course can be accessed from here.

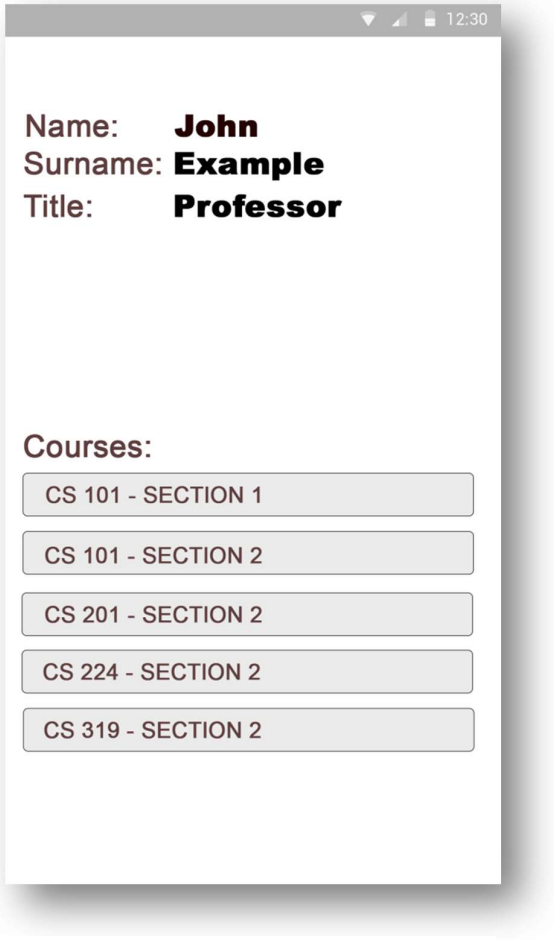



*Figure 3 - The list of the courses*

### 3.4.5.2.3 Courses Screen

The following image shows the interface of a particular course. From top bottom the name of the respective course in a green-red bar is shown. The green bar represents the percentage of the students present and the red represents those absent. Some information regarding the specific related numbers is shown just below the course code adjacent to the date.

Below, the grid shows the pictures of the students and a ‘tick’ mark representing they are present and an ‘X’ marking those absent. There will also be a search bar to search for a specific student.

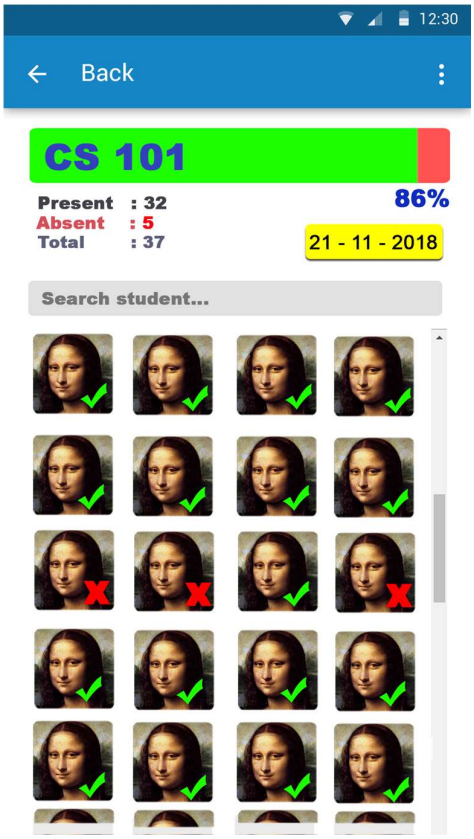



*Figure 4 - Course screen / professor*

#### 3.4.5.2.4 Objections Screen

The following screen shows the list of the objections made by students regarding attendance that was not picked by the system. The professor has to decide whether to mark the student present or not. This decision is made by looking by the picture that the system took to mark the students, shown in details in next section.


*Figure 5 - Objections Screen*

### 3.4.5.2.5 Objections Screen 2

The following screen is the continuation of the previous section. From the previous section the professor can click on the student objection and will directly be shown the respective class picture taken by the system. All of the persons in the picture are marked with rectangles. As seen in the picture all of them are recognized except for one. The professor here has to click on the rectangle corresponding to the face of the student not recognized and the student will be marked present. This manual work done by the professor will be used further by the system to train the algorithm.

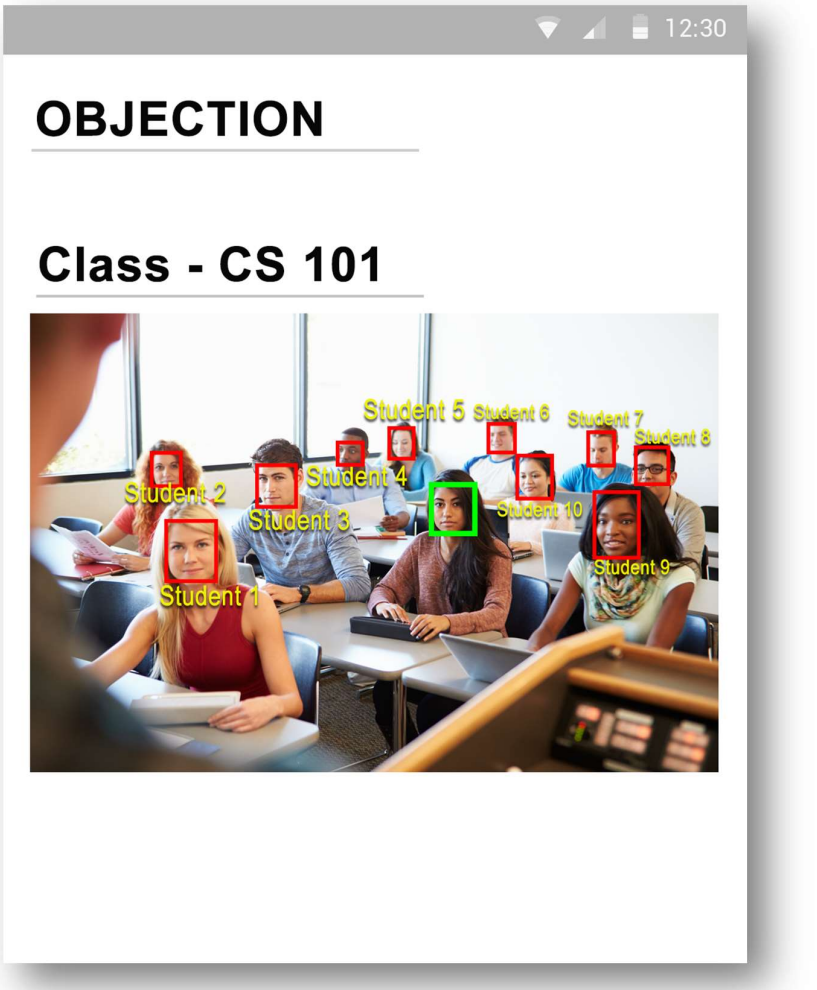



*Figure 6 - Second Objections Screen*

#### 3.4.5.2.6 Settings Screen for Professors

The following screen shows the settings that can be changed by the professor. Basically the professor may choose to make the notification sound or silent. Also the professor may choose to get a notification when the class attendance reaches a very low value.

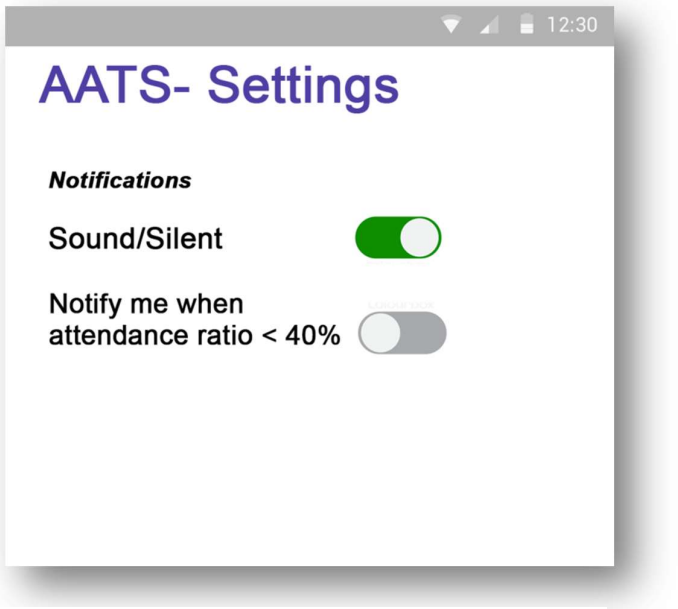



*Figure 7 - Settings Screen / Professors*

### 3.4.5.3 User Interface for Students

The following user interface sections correspond to the side that the students will be seeing in their application.

#### 3.4.5.3.1 Main Screen Students

The picture shows the main screen for the students where personal information is as shown. Note that the student can access the list of the courses they are taking and the settings.

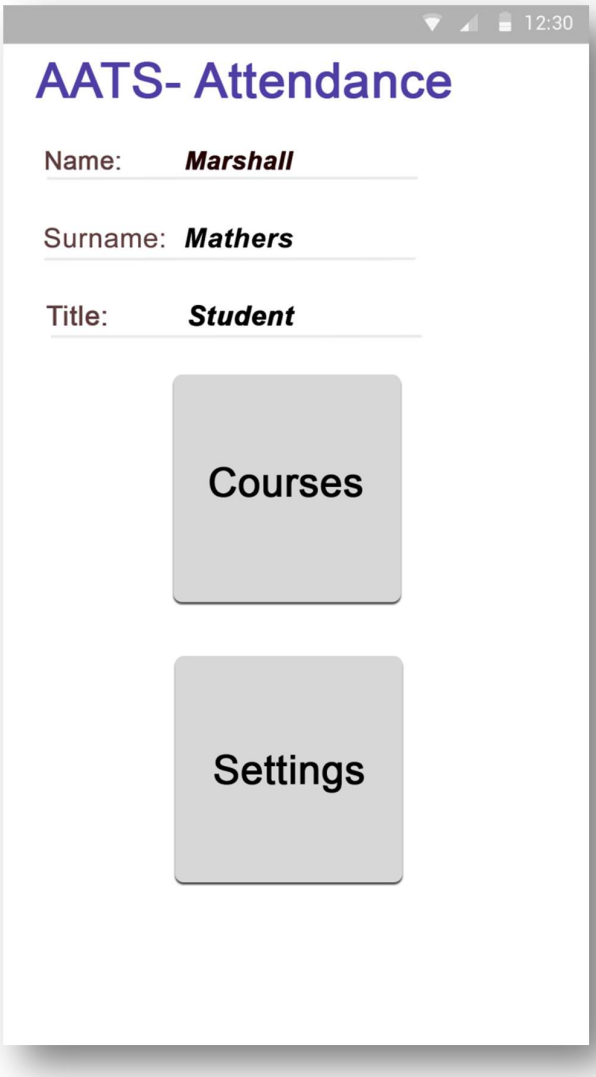



*Figure 8 - Main Screen / Student*

#### 3.4.5.3.2 List of Courses Screen Student

The following screen shows the list of the courses taken by the student as well as the student's personal information.

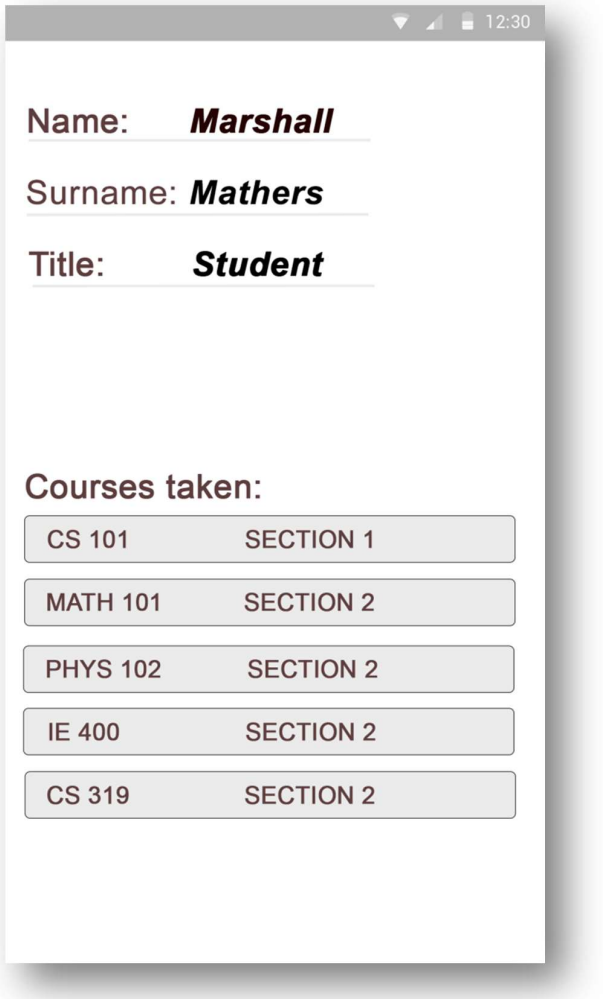



*Figure 9 - List of Courses / Student*

### 3.4.5.3.3 Course Attendance Information

The following screen shows the course code adjacent to the list of days and the corresponding attendance taken.

The student can make an objection if the attendance for a particular day is not taken correctly. However this option is only available at the end of the course and will expire instantly 10 minutes after the classes finish.

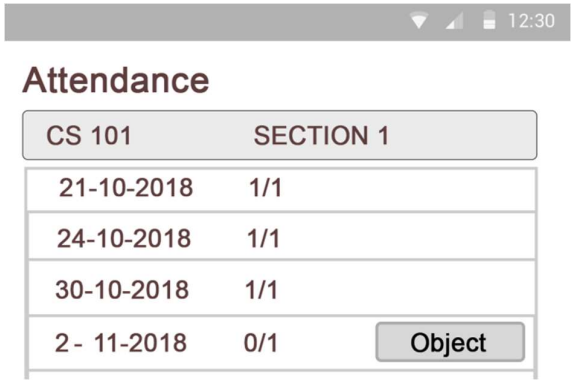



*Figure 10 - Course Attendance Student*

#### 3.4.5.3.4 Settings Student Version

The following screen shows the settings that the student can change. Every student can change the notification type to sound or silent. Also the student has the option of getting notified when the system marks them absent. This is to ensure that the students make objections inside the time limit when the system doesn't recognize them. The student must attend to the professor right away as the class finish and address the respective objection in case one was made.

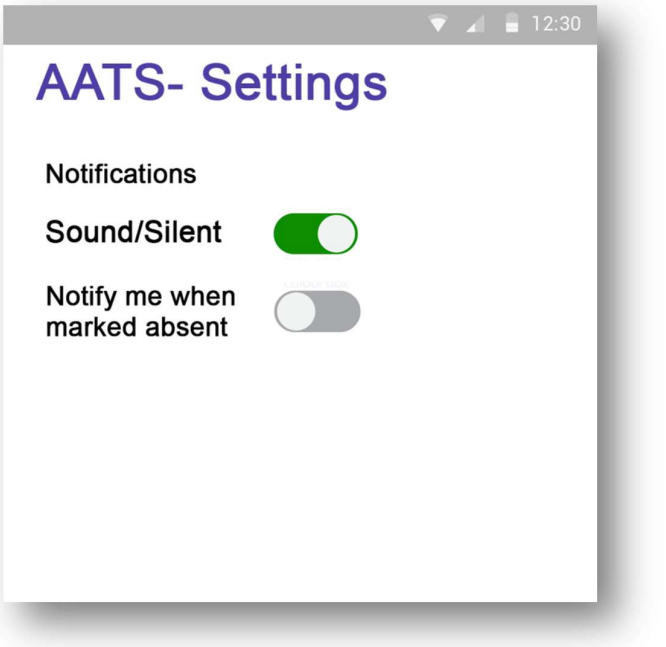



*Figure 11 - Settings Screen / Student*

## 3. References

[1] https://www.engineerica.com/accuclass

[2] https://www.myattendancetracker.com