Point Blank Consultants

Damian Stenzel 718761

Nureen Khan 674169

Maggie-May Hosking 841942

Lindsay John Arendse 800648

Jason Chalom 711985

|  |
| --- |
| **Milestone 1: Analysis, Database and User Interface Design**  Group 10: Point Blank |

# 1. Executive Summary

Based on the feedback received from the previous semester we have made necessary changes to the document. We have revisited our use case set and based on the changes in requirements have edited this appropriately. The following document contains numerous models which breakdown the system functionality.

After having revised Team 1, Team 9 and Team 11’s use case set as well as the updated requirements provided in the project case study we have compiled a comprehensive list of use cases.

We have included the required fully dressed use cases, being “CreateStudent”, “CreateStudent\_Attendance” and “UpdateStudent\_Attendance”. These fully-dressed use cases provide a brief description of the scope, related actors, use cases, flow of activities and the pre and post conditions.

Given the changes in the case study we have edited the state machine diagram accordingly to include point ranges, which determine when and how a student will be ranked. This will serve as an element of our implementation of gamification.

With the addition of tables and attributes in our database we have had to reconfigure our class diagram to accommodate these changes. The class diagram is the basis of our relation database model, which our entity relationship diagram is also derived. The class diagram clearly illustrates what data is needed in order to support the execution of the business’ processes required for this system. The class diagrams also show all relationships between objects and the flow of data. This is the basis for the business rules described in the given case study.

A data dictionary has been attached as appendix A. It is a further in-depth description of the entities and their attributes from the database described in the entity relationship diagram.

Our team design guidelines state what specifications we would like to include in the user interface. These guidelines will help to ensure consistency across our application and when designing the application in Visual Studio.

Storyboards for “Create Student” and “Rate Tutor” have been included. This is gives a representation of the navigational flow and functionality of the two use cases respectively. These have been done in accordance with the fully dressed use cases. (Note: The fully dressed use case for “Rate Tutor” has been included as Appendix B.)

# 2. Revised Analysis

## 2.1 Revised Use Case Set

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Source | Use Case | Event Classification: | Event | Trigger | Response | Destination |
| Student | Rate Tutor | External | Student wants to rate tutor | Student selects rate tutor | Tutor is rated | System database |
|  | Rate activity | External | Student wants to rate an activity | Student selects rate activity | Activity is rated | System Database |
|  | Create student attendance | External | Student wants to register for activities | Student selects create student attendance | Student is registered to attend activities | System database |
|  | Update Student | External | Student wants to update their profile | Student selects update profile | Student profile is updated | System database |
|  | Create Consultation | External | Student wants to create consultation with a tutor | Student selects create consultation | Consultation is created | System Database |
| SSS Coordinator | Create student | External | SSS coordinator wants to register a student | SSS coordinator selects register student | Student is registered | System database |
|  | Create Tutor | External | SSS coordinator wants to create tutor | SSS Coordinator selects create coordinator | Tutor is created | System Database |
|  | Update student | External | SSS coordinator wants to update student | SSS coordinator selects update student | Student is updated | System database |
|  | Update tutor | External | SSS coordinator wants to update tutor | SSS coordinator selects update tutor | Tutor is updated | System Database |
|  | Assign tutor | External | SSS coordinator wants to assign a tutor to an activity | SSS coordinator selects assign tutor | Tutor is assigned | System Database |
|  | Generate report | External | SSS coordinator wants to obtain information about students, tutors and activities | SSS coordinator prompts system for support and offerings analysis. | System generates a detailed report based on information that has been captured. | System database |
| SSS tutor | Update consultation | External | SSS tutor wants to update consultation | SSS tutor selects update consultation | Consultation is updated | System Database |
|  | Update activities | External | SSS tutor wants to update activity | SSS tutor selects update activity | Activity is updated | System Database |
|  | Update tutor | External | SSS tutor wants to update profile | SSS tutor selects update profile | Tutor is updated | System Database |
|  | Update student attendance | External | SSS tutor wants to update student attendance | SSS tutor Selects Update student Attendance | Student attendance is updated | System database |
|  | Update Student Consultation | External | SSS tutor wants to update a student consultation | SSS tutor Selects Update student consultation | Students consultation is updated | System Database |

## 2.2 Revised “Create student” Use Case

|  |  |  |
| --- | --- | --- |
| Use case name: | CreateStudent | |
| Scope: | Student Support System(SSS) | |
| Triggering event: | SSS coordinator initiates create student (i.e. when the SSS co-ordinator clicks “create student”) | |
| Brief description: | The SSS coordinator initialises the system to create a new student. The SSS coordinator then enters in student details, namely: student number, first name, last name, ID or passport number, date of birth, e-mail address, mobile number, year of study, degree programme, registered courses etc. System will automatically set student status to “white”. Once registered, the system will then automatically generate and send an e-mail to the student to confirm that he/she has been registered for the programme and to remind the student that he/she can log into the system with immediate effect and begin to sign up for activities. Login credentials will also be provided, the username being the student number and the password being system-generated. | |
| Actor(s): | Primary: SSS coordinator | |
| Related use cases: | N/A | |
| Stakeholders and interests: | SSS coordinator: Wants to register students. SSS coordinator wants accurate and fast data capture.  Student: Wants to be registered in a fast streamline manner, and receive a confirmation stating a successful registration. | |
| Pre-conditions: | N/A | |
| Post-conditions: | Student Status assigned to white  Student must be created in the student datastore  Generate password  Store username in student datastore  Store password in student datastore  Automatically send confirmation email to student  Student confirms registration | |
| Flow of activities: | Actor   1. Coordinator requests to create student 2. Coordinator enters in required student details 3. Coordinator confirms registration | System   * 1. Prompts coordinator to enter student number   1.2 Prompts coordinator to enter student ID or passport  1.3 Prompts coordinator to enter student first name  1.3 Prompts coordinator to enter  student last name  1.4 Prompts coordinator to enter student date of birth  1.5 Prompts coordinator to enter student e-mail address  1.6 Prompts coordinator to enter mobile number  1.7 Prompts coordinator to enter student year of study  1.8 Prompts coordinator to enter student degree programme  2.1 Set student status to “white”  2.2 Prompt coordinator to confirm student registration  3.1 Set user name to student number  3.2 Generate password  3.3 Store student in student data store  3.4 Generate confirmation of registration email  3.5 Display student registration successful  3.6 Display confirmation email sent successful |

## 2.3.1 “Create student\_attendance” Use Case

|  |  |  |
| --- | --- | --- |
| Use case name: | CreateStudent\_Attendance | |
| Scope: | SSS System | |
| Triggering event: | When student clicks “Register for activities” | |
| Brief description: | Students will register once-off to enable themselves to attend activities as part of a group. These activities will run in multiple parallel sessions | |
| Actor(s): | Primary: Student | |
| Related use cases: |  | |
| Stakeholders and interests: | SSS co-ordinator, to monitor how many students have registered.  SSS tutor, to know how many students plan to attend an activity.  Student, to ensure that he/she has registered their attendance for activities. | |
| Pre-conditions: | Student must exist.  Groups must exist. | |
| Post-conditions: | Student must be registered as part of a group in order for him/her to attend activities. | |
| Flow of activities: | Actor   1. Student requests to register for activities. 2. Student selects a group.   . | System   * 1. Prompts Actor to select a group to join.   2.1 Student is added to activity attendance list for requested group.  2.2 Display successful registration for activities |

## 2.3.2 “Update student\_attendance” Use Case

|  |  |  |
| --- | --- | --- |
| Use case name: | UpdateStudent\_Attendance | |
| Scope: | SSS System | |
| Triggering event: | When the SSS tutor clicks “Update Student Attendance” | |
| Brief description: | SSS tutor may update student attendance by assigning each student who attended their activity “Present”. | |
| Actor(s): | Primary: SSS tutor | |
| Related use cases: | CreateStudent\_Attendance | |
| Stakeholders and interests: | SSS co-ordinator to monitor student attendance for activities and consultations.  Students to track attendance and to ensure SP requirements are met.  Lecturers to identify students at risk. | |
| Pre-conditions: | Student must exist | |
| Post-conditions: | Student attendance is updated and marked as either “Absent” or “Present” for each specific activity of consultation. | |
| Flow of activities: | Actor   1. SSS tutor requests to update student attendance. 2. SSS tutor selects which activity to update. 3. SSS tutor updates student attendance accordingly. | System   * 1. Prompts Actor to select which activity he/she wishes to update.   2.1 Prompts actor to mark student attendance as “Present”.  3.1 Display Update Successful |

## 2.4 Revised state machine diagram for the STUDENT object

The above diagram illustrates the possible states a student can reach in the SSS programme through participation. Each activity outlined in the student handbook has a set amount of points allocated to it. Upon registration each student will be automatically set to a white status. In order to progress to “BLUE” a student will have to have accumulated a minimum of 50 points, through participation of various activities. The same concept applies to all states, an accumulation of 100 points triggers a state move to “BRONZE”, a further accumulation of 50 points, putting a student at a total of 150 triggers a state move to “SILVER” , once a student has reached 170 points they have reached the highest level of achievement being “GOLD”. As set out in the student handbook should a student miss a scheduled activity, 10 points will be deducted from the running total; this may cause a student to be demoted. The use of intervals depicts the various ranges determining at which points the student will be demoted.

## 2.5 Revised Class Diagram



The above class diagrams is a visual depiction showing what data needs to be stored in the respective data stores and shows associations and relationships between different objects. In the SSS class diagram the things (or objects) are Student, Tutor, Tutorial and SSS coordinator. Each object has attributes which define the object (e.g. a Student has a first name and last name). These objects are what trigger and interact with the SSS system. Between these objects there are relationships which define how the system will work, these are based on and extracted from your business rules.

### 2.5.1 Business Rules for class Diagram:

* A student attends zero to many consultations and a consultation is attended to one and only one student.
* A student is registered by one and only one SSS coordinator, a SSS coordinator can register zero to many students.
* A student attends zero to many activities, student activity comprises of one to many students.
* A tutor facilitates zero to many consultations, a consultation is attended by one and only one student.
* A tutor facilitates zero to many activities, and an activity is facilitated by one and only one tutor
* A tutor facilitates zero to many consultations and consultations are attended by one and only one student.
* Note: The association class indicated by the dashed line in the above picture enables you to identify a specific student’s attendance for a specific activity.

## 2.6 Entity Relationship Diagram (ERD)

## 

### 2.6.1 The purpose of the Entity Relationship Diagram:

An entity-relationship diagram (ERD) is a graphical representation that shows the relationship between people, objects, concepts or events within an information system. This model helps us define business processes and can be used as the foundation for a relational database.

* A student attends zero to many consultations and a consultation is attended to one and only one student.
* A student is registered by one and only one SSS coordinator, a SSS coordinator can register zero to many students.
* A student attends zero to many activities, student activity comprises of one to many students
* A tutor facilitates zero to many consultations, a consultation is attended by one and only one student.
* A tutor facilitates zero to many activities, and an activity is facilitated by one and only one tutor
* A tutor facilitates zero to many consultations and consultations are attended by one and only one student.
* The many-to-many relationship has been resolved through the addition of the attendance table which identifies a specific student’s attendance for a specific activity.

# 3. Data Dictionary

The data dictionary is a further in-depth description of the entities and their attributes from the database described in the Entity Relationship Diagram (ERD). The data dictionary for this project is located in Appendix A at the end of this document.

# 4. Design Guidelines

|  |  |
| --- | --- |
| Font – Default Text | Calibri – size 12 |
| Font – Headings | Calibri – size 14 |
| Colour Scheme: Foreground (Text) | Black |
| Background | White |
| Border | Blue |
| Navigation | * Buttons will be placed at the bottom of the screen, guiding users through the program. * A menu will be included on the left hand side of the screen, showing the user various options of the program as well as mimicking that of an audit trail by highlighting the users current position on the menu (in blue). This will be linked to the navigation buttons at the bottom of the screen.   Direct manipulation will be enforced so that users are working directly with the object. |
| Title and Field Labels | Will be named in accordance with our use case set and ERD. Please note this will not be a perfect match, they will however resemble the naming conventions prescribed in the above mentioned models. |
| Icons | Icons will be included, in order to enforce double coding e.g. Date of birth will be accompanied by a calendar. |
| Screens | Will abide by the chunking rule of +/- 7 groups of smaller information(In order to prevent over population) |
| Error Messages | Will appear in red as tooltips for missing/invalid data in the appropriate field. |
| Confirmation/Success Messages | Will appear as pop up boxes they will contain a heading specifying “Success!” informing the user of what they have successfully completed. |

# Design Guidelines Reasoning:

**Font** – We selected Calibri as our font of choice as it is the default font setting for Windows and we feel that it is best to keep to a font which the user is familiar with.

**Colour scheme** – At this point we have decided to keep our colour scheme fairly standard, by choosing black as foreground and text, white for the background and having a blue boarder. This is once again in keeping with something that users are familiar with and not too bold or overwhelming.

**Navigation** – We feel that it is most convenient to have navigation buttons positioned at the bottom of the screen as users read from top to bottom of the page.

Our team would like to develop the idea of a side menu which mimics an audit trail, we feel that this will help the user in the navigation process and will make it easier for the user to switch from one section of the application to another.

**Titles and Field Titles** – Titles will be kept simple and will follow in accordance with our use case set and ERD, this is in an attempt to insure that no fields are left out and to insure that all necessary data will be captured.

**Error Messages** – Our choice to indicate error messages using tooltips is to prevent bombarding the user with popup messages. With the use of tooltips, we are also able to point out to the user exactly where the error has occurred.

**Confirmation/ Success Messages** – We have chosen to use pop up boxes containing messages in them to make the user aware of their success, which we feel will make them more willing to continue using our application as they can see their progress.

# 5. Storyboards

## Notes:

1. This story board wireframe and Example form is a provisional design which may change in the future.
2. The wireframe storyboards do not represent the designs of each aspect but rather the user interaction and flow.
3. The design guidelines specified above will be essential to our final design later on in the project.
4. These story boards are used as only a representative example of the user interaction flow. These flows within each screen will stay the same but the specific designs of these screens are subject to change. For example the menu bar for both story boards has been used as a temporary representative example to show how we wish to have an audit trail however our design guidelines specify using a sidebar instead of a menu bar.
5. The starting and ending on the home page of the system is only there for user flow. The user may start and end on many different pages of the application.
6. This flow may only be done by the course coordinator
7. A form initiator is some element or object on our design which allows the user to initiate a flow or process from the specific screen they are on. This initiator can be a menu object, button or some other user interface interactive device.
8. Appendix B contains the Rate Tutor fully-dressed use case for 5.2

## 5.1 Create Student Use Case



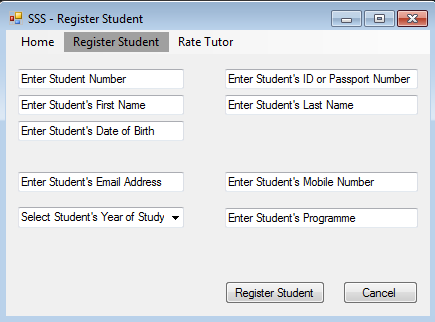
### Flow:

1. “The happy path” case has been assumed in this representative example.
2. The user is assumed to start on the home page and click on the register student form initiator.
3. Once the register student form is open, the user will fill in the required fields and press enter.
4. The user may also cancel by closing the window with the default windows components or by pressing the cancel button.
5. The data to input on the form has been chunked together according to what information is related.
6. Validation for this part of the flow is discussed below.
7. Once the process has been completed a status message dialogue is displayed to the user.
8. Once the user has closed the status message dialogue they are returned to the screen they came from. In this case the home screen.

### Validation:

1. Student Number must be entered
2. Student ID and Passport number entered
3. Students first name must be entered, string only
4. Students last name must be entered, string only
5. Students date of birth must follow (YYYY-MM-DD)
6. Student email address is entered
7. Students mobile number must be entered and numeric
8. Year of study must be selected from a drop down list
9. Students programme must be entered

### 5.1 Example Form



## 5.2 Rate Tutor Use Case



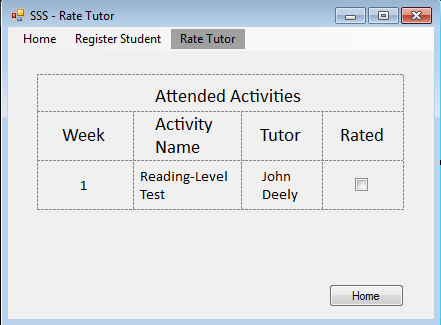
### Flow:

1. “The happy path” case has been assumed in this representative example
2. The user is assumed to start on the home page and click on the rate tutor form imitator.
3. The user selects from the list of attended activities, the activity which the user wants to rate the tutor.
4. The user selects from 5 radio buttons the rating of their choice for the tutor, 1 being “poor” and 5 being ”excellent” .
5. Once the process has been completed a status message dialogue is displayed to the user.
6. The user is then sent back to the rate tutor page
   1. The rating check box is now checked
   2. **Note:** the rating checked is a derived value from checking if the rating for the activity stored in the database is null or has a value

### Validation:

1. Only activities that were attended will appear on the “Attended Activities” list.
2. Tutor rating must be selected. i.e. There will be no default selection on tutor rating.

### 5.2 Example Form



# Appendix A: Data Dictionary

## **Table:** CONSULTATION

**Scope:** Data about a consultation, this includes a consultation number, tutor number as well as a student number. The table stores the

attributes of objects belonging to the class “Consultation”.

**Attribute Listing:**

**Consultation\_No**

Description: A unique consultation number which uniquely identifies each consultation

Type: Auto increment (integer)

Format: e.g. [7] which will uniquely identify the 7th created consultation

Note: Primary key

**Coordinator\_No**

Description: A unique coordinator number which uniquely identifies each coordinator

Type: Auto increment (integer)

Format: e.g. [4] which will uniquely identify the 4th created coordinator

Note: Foreign Key used to define the relationship between the CONSULTATION table and the SSS COORDINATOR table

**Student\_No**

Description: A unique student number which uniquely identifies a student

Type: Auto increment (integer)

Format: e.g. [4] which will uniquely identify the 4th registered student

Note: Foreign Key used to define the relationship between the CONSULTATION table and the STUDENT table

**Consultation\_Details**

Description: will store any important details or notes about the consultations

Type: Text

Format: e.g. “It was decided that student 4 will be granted permission to catch up missed activity due to personal issues”

Note:

**Consultation\_Date**

Description: Will store date of consultation

Type: Date

Format: YYYY-MM-DD

Note:

**Student\_Arrived**

Description: States whether a student arrived for their consultation

Type: Boolean

Format: Y/N

Note:

## Table: TUTOR

**Scope:** Data about tutors, this includes a tutor number, as well as a coordinator number. The table stores the

attributes of objects belonging to the class “Tutor”.

**Attribute Listing:**

**Tutor\_No**

Description: A unique tutor number assigned by which uniquely identifies each tutor

Type: Auto incremented (integer)

Format: e.g. [4] which will uniquely identify the 4th registered tutor

Note: Primary key of TUTOR table

**Tutor\_FirstName**

Description: Will store tutor first name

Type: String

Format: N/A

Note:

**Tutor\_LastName**

Description: Will store tutor last name

Type: String

Format: N/A

Note:

**Tutor\_Id\_Passport**

Description: Will store tutor id/passport number

Type: Integer

Format: N/A

Note:

**Tutor\_DateOfBirth**

Description: Will store tutor date of birth

Type: Date

Format: YYYY-MM-DD

Note:

**Tutor\_EmailAddress**

Description: Will store tutor email address

Type: Alpha/Numeric

Format: N/A

Note:

**Tutor\_CellNumber**

Description: Will store tutor cell phone number

Type: Integer

Format: (###) ### ####

Note:

**Tutor\_TrainingStatus**

Description: Will store tutor training status

Type: Boolean

Format: Y/N

Note:

## **Table:** STUDENT

**Scope:** Data about students, this includes a student number, as well as a coordinator number. The table stores the attributes of objects belonging to the class “Student”.

**Attribute Listing:**

**Student\_No**

Description: A unique student number assigned by which uniquely identifies each student

Type: Auto incremented (integer)

Format: e.g. [4] which will uniquely identify the 4th created student

Note: Primary Key of student table

**Group\_No**

Description: The number assigned to the group that a student is registered to

Type: Integer

Format: N/A

Note: Foreign Key of Student table, links the student table and the group table

**Coordinator\_No**

Description: A unique coordinator number which uniquely identifies each coordinator

Type: Auto increment (integer)

Format: e.g. [4] which will uniquely identify the 4th created coordinator

Note: Foreign Key used to define the relationship between the STUDENT table and the SSS COORDINATOR table

**Student\_FirstName**

Description: Will store student first name

Type: String

Format: N/A

Note:

**Student\_LastName**

Description: Will store student last name

Type: String

Format: N/A

Note:

**Student\_Id\_Passport**

Description: Will store student id/passport number

Type: Integer

Format: N/A

Note:

**Student\_DateOfBirth**

Description: Will store student date of birth

Type: Date

Format: YYYY-MM-DD

Note:

**Student\_EmailAddress**

Description: Will store student email address

Type: Alpha/Numeric

Format: N/A

Note:

**Student\_CellNumber**

Description: Will store student cell phone number

Type: Integer

Format: N/A

Note:

**Student\_YearOfStudy**

Description: Will store student year of study

Type: Integer

Format: N/A

Note:

**Student\_DegreeProgramme**

Description: Will store student Degree programme

Type: String

Format: N/A

Note:

**Student\_Status**

Description: Will store student status

Type: \*\*\*\*\*\*\*\*\*

Format: N/A

Note:

**Student\_Points**

Description: Will store student points

Type: Integer

Format: N/A

Note:

## **Table:** SSS COORDINATOR

**Scope:** Data about SSS Coordinator, this includes a Coordinator number. The table stores the attributes of objects belonging to the class “SSS Coordinator”.

**Attribute Listing:**

**Coordinator\_No**

Description: A unique coordinator number assigned by which uniquely identifies each coordinator

Type: Auto incremented (integer)

Format: e.g. [4] which will uniquely identify the 4th created coordinator

Note: Primary Key of SSS coordinator table

**Coordinator\_FirstName**

Description: Will store Coordinator first name

Type: String

Format: N/A

Note:

**Coordinator\_LastName**

Description: Will store Coordinator last name

Type: String

Format: N/A

Note:

**Coordinator\_Id\_Passport**

Description: Will store Coordinator ID/Passport number

Type: Integer

Format: N/A

Note:

**Coordinator\_DateOfBirth**

Description: Will store Coordinator date of birth

Type: Date

Format: YYYY-MM-DD

Note:

**Coordinator\_Emailaddress**

Description: Will store Coordinator email address

Type: Alpha/Numeric

Format: N/A

Note:

**Coordinator\_Cellnumber**

Description: Will store Coordinator cell phone number

Type: Integer

Format: N/A

Note:

## **Table:** STUDENT\_ACTIVITY

**Scope:** Data about student activities, this includes a student activity number. The table stores the attributes of objects belonging to the class “Student Activity”.

**Attribute Listing:**

**Student\_Activity\_No**

Description: A unique student activity number assigned by which uniquely identifies each student activity

Type: Auto incremented (integer)

Format: e.g. [4] which will uniquely identify the 4th created student activity

Note: Primary Key of Student Activity table

**Student\_Activity\_Name**

Description: Will store the student activity name

Type: String

Format: N/A

Note:

**Student\_Activity\_PointValue**

Description: Will store the student activity point value

Type: String

Format: N/A

Note:

**Student\_Activity\_Week**

Description: Will store the student activity week

Type: String

Format: N/A

Note:

**Student\_Activity\_Description**

Description: Will store the student activity description

Type: String

Format: N/A

Note:

## **Table:** ATTENDANCE

**Scope:** Data about student activity attendance, this includes a student activity number, as well as a student number and tutor number. The table stores the attributes of objects belonging to the class “Student Activity Attendance”.

**Attribute Listing:**

**Student\_Activity\_No**

Description: A unique student activity number assigned by which uniquely identifies each student activity

Type: Auto incremented (integer)

Format: e.g. [4] which will uniquely identify the 4th created student activity

Note: Primary Key of Student Activity Attendance table

**Student\_No**

Description: Will store the student number

Type: Integer

Format: N/A

Note: Foreign Key used to define the relationship between the STUDENT ACTIVITY table and the STUDENT ACTIVITY ATTENDANCE table

**Tutor\_No**

Description: Will store the tutor number

Type: Integer

Format: N/A

Note: Foreign Key used to define the relationship between the TUTOR table and the STUDENT ACTIVITY ATTENDANCE table

**Student\_Activity\_TutorRating**

Description: Will store the tutor ratings

Type: Integer

Format: N/A

Note:

**Student\_Activity\_Rating**

Description: Will store the Activity Rating

Type: Integer

Format: N/A

Note:

**Student\_Arrived**

Description: States whether a student arrived for their consultation

Type: Boolean

Format: Y/N

Note:

## Table: GROUP

**Scope:** Data about the group that students are assigned to in order to attend activities , this includes a group number, as well as group dates and times. The table stores the attributes of objects belonging to the class “Group”.

**Attribute Listing:**

**Group\_No**

Description: A unique group number which corresponds to each activities date and time.

Type: Integer

Format: N/A

Note: Primary Key of group table

**Group\_Date**

Description: Identifies activity date

Type: Date

Format: YYYY-MM-DD

Note:

**Group\_Time**

Description: identifies activity time

Type: Time

Format: hh:mm

Note:

# Appendix B: RateTutor Use Case Description

|  |  |  |
| --- | --- | --- |
| Use case name: | RateTutor | |
| Scope: | Student Support System(SSS) | |
| Triggering event: | SSS Student clicks “Rate Tutor” | |
| Brief description: | The student wishes to rate the Tutor, which they have attended. With a rating from 1-5, 1 being poor and 5 being excellent | |
| Actor(s): | Primary: student | |
| Related use cases: | CreateActivity, UpdateAttendance, CreateAttendance, | |
| Stakeholders and interests: | Student: students gain points through ratings  Tutor: Tutors will be able to see their performance  SSS Coordinator: SSS coordinators will be able to track tutor ratings | |
| Pre-conditions: | Activity has to have been attended | |
| Post-conditions: | Tutor will have been rated  Student will have gained points | |
| Flow of activities: | Actor   1. Student selects rate Tutor 2. Student selects activity 3. Student rates tutor | System   * 1. System prompts user to choose from a list of attended activities   2. Prompts user for tutor rating      * 1. Invoke UpdateActivity   2. Display successful rating of tutor |

Note: This Fully-dressed use case is used for the one story board (5.2)