RoboOfficeNXT System Requirement Specification

**Student Management Module**

**Entity Definition**

**Student** – contains personal information related with one unique student.

**Student’s status** –

Active – Student has at least 1 active progress.

Inactive – Student has no active progress.

Prospective – Student has no progress.

Active/Inactive status can be toggled by user. However, prospective status is controlled by the system.

**Progress** – contains a student’s single course/membership information, it is dynamic and subjected to change as the student progresses into the course/membership.

**Progress’s status** – Active, Inactive, Suspended

Active: set by user. Default after new payment

Inactive: set by user

Suspended: this state can only occur during active stage. A suspended or inactive progress will not appear on daily attendance.

**Class/Club record** – contains a student’s activity of a particular time and a particular ongoing progress. Normally each record has at least one project associated with it, and only one time slot.

**Timeslot** – indicates a timestamp of a class start time. It is composed of a particular day of the week + a start time. Ex: Monday 14:00, Wednesday 16:00, etc.

**Daily Attendance** – List of student progresses, ordered by timeslots then classrooms. Used by teacher and receptionist. Progress must be within date range and active.

* **Enter Course attendance Scenario**

In general, If progress status = inactive, make it active

Progress ++

|  |
| --- |
| If progress == total |
| * Check for next progress, if available   **Next progress**   * Status = active * Classrecord’s progress id = next progress id   **Current progress**   * Status = inactive * Available credit = available credit – (available credit / total) * If next progress not available   **Current progress**   * Status = inactive * Available credit = available credit – (available credit / total) |
| If progress < total |
| * Current Progress * Status = active * Available credit = available credit – (available credit / total) |
| If progress > total   * Current progress * Status = active * Available credit = available credit – (available credit / total) |

**Enter Membership Attendance**

Workshop teacher writes down project name along with attendance. Office enters projects.

Course and Membership Attendance Form format:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Course name | Student name | Progress/total | signature | Project |

Example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Robokids 1 | Jason Zhang | 4/12 | Jason zhang | Dancing bird |

Each classroom has its own Excel Sheet

Each excel sheet has at least one timeslot with progresses under each timeslot

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Jan/13/2013 Sat 14:00** | | | | |
| **Robokids 1** |  |  |  |  |
| **Robokids 2** |  |  |  |  |
| **Robokids 3** |  |  |  |  |
|  |  |  |  |  |

Attendance sheet includes only student with active status and progresses that are marked active.

**Record Attendance:**

Mark students that attended. Allow add student progress: Search student, show student + course progress dropdown list.

**Enter progress report:**

Teachers login and fill incomplete record as complete.

Fields include: Project name

If course record: progress report for this particular record.

**Progress report format:**

9 scores + 4 comments, 2 comments are mandatory

is a test / or not

**Report Card Calculation**

9 Criteria score: calculated using average of criteria \* (3/max score)

Example: average of building-design = 4.5 out of 5, then final score = 4.5 \*3 / 5 = 2.7

0~1 below average, 1~2 average, 2~3 = above average

2.7 = above average

Note: When calculating average, tests has weight of 2. Others have weight of 1.

For each report card, three section comment + behavior comment needs to be filled by teacher, 3 sections are filled based on score, using strength/weakness + recommendation template.

**Finance Module**

**Entity Definition**

**Payment –** Each payment has at least one payment entry.

Each payment has a unique invoice number.

Each payment has a notes attached for office used, that can be empty.

Each payment has received by user tag + a date paid.

Each payment is subject to only one student.

Each payment has a total that is the sum of all subtotals of each payment entry.

**Tax –** HST/ GST/PST/GST+PST

**Payment Entry** – Each payment entry has a type associated.

Each payment entry has an initial amount.

Each payment entry has a discount section.

Each payment entry may or may not be subjected to tax

Each payment entry is calculated with the following formula:

Subtotal = (initial amount – discount) \* (1+tax)

Discount Section: contains a discount amount + a discount description.

Discount description cannot be empty if discount amount > 0

* **FR 1.1.0 – make payment in general**

Receptionist selects a current student (Active, Inactive, or Prospective), and make a payment to this student. Each payment contains at least one payment entry, which includes Courses, Memberships, and Other fees.

After payment, student status is active.

During any instance, only one payment can be made. Two payment cannot be entered at the same time.

* **FR 1.1.1 – add Payment Entry, new Course.**
* **FR 1.1.2 – add Payment Entry, new Membership**
* **FR 1.1.3 –** **add Payment Entry, new Fees.**

**FR 1.1.1 – add Course Payment Entry**

* 1. PAY NEW COURSE: New course is independent of all other current progresses.

Normally applies to new student, or existing student who wishes to continue in another field. (Ex, Robokids students who wants to take RoboMath course)

* 1. PAY IN ADVANCE: New course starts when a current course progress is finished.

Normally occurs when a parent wants to pay in advance, when the current progress has 1 or 2 weeks left (11/12 progress).

If current progress has overdue class (13/12 progress), the overdue part is carried toward the new course.

* 1. LEVEL-UP PAY: the current course progress credit is used in the payment of the new course.
  2. PAY OVERDUE AMOUNT: If a current course progress has overdue record, (8/6), then pay the overdue part only (which is 2 classes, in this case).

**FR 1.1.2 – add Payment Entry, new Membership**

PAY NEW MEMEBERSHIP

If the membership being paid has already started, then the student pays for the remaining amount, calculated using the number of days till end date.

**FR 1.1.3 –** **add Payment Entry, new Fees.**

Add a particular fee to the student. Fees can be selected from Pre-defined list, or can be customized.

|  |
| --- |
| PAY NEW COURSE |
| Initial amount = New course rate \* ( weeks – discount class)  Newly created course progress:   * Available credit = amount paid before tax * Default start date = today * Progress = 0 * section timeslot changed = 0 * bonus class = 0 * next progress = null * modified date = payment date * status = active |

|  |
| --- |
| PAY IN ADVANCE |
| Initial amount = new course rate \* (weeks – discount class)  If last course progress : progress <= total (Not Overdue)   * Newly created course progress: * Default start date = week after the end date of the last course progress * Default timeslot = last course progress (if available for the course chosen) * Available credit = amount paid before tax * Progress = 0 * Section timeslot changed = 0 * Bonus class = 0 * Next progress = 0 * status = inactive * Modified date = payment date * Last course progress * Next progress = newly created course progress   If progress >= total (Last course progress overdue)   * Newly created course progress * Default start date = first overdue class record * Default timeslot = last course progress (if available for the course chosen) * Available credit = amount paid before tax * Progress = number of overdue record * Section timeslot changed = 0 * Bonus class = 0 * Next progress = 0 * Status = active * Modified date = payment date * Last course progress * Next progress = newly created course progress * Progress = total * Status = inactive * Modified date = payment date * Modified user = system * Overdue Class record * Update progress id to newly created progress id * Modified date and user = system and date |

|  |
| --- |
| LEVEL-UP PAY |
| Initial Amount = new course rate \* (weeks – discount class) – last progress credit   * Newly created course progress: * Default start date = week after the end date of the last course Record of the last course progress * Default timeslot = last course progress (if available for the course chosen) * Available credit = amount paid before tax * Progress = 0 * Section timeslot changed = 0 * Bonus class = 0 * Next progress = 0 * status = active * Modified date = payment date * Last course progress * Next progress = newly created course progress * Status = inactive * Total class = progress * Modified date and user = system |
| PAY OVERDUE AMOUNT |
| Initial Amount = old course rate(from last payment record) \* (weeks – discount class)  (weeks must >= number of overdue classes)   * Existing Course progress * Total class += weeks * Available credit = available credit + amount paid – overdue classes \* rate |

|  |
| --- |
| PAY NEW MEMBERSHIP |
| Initial Amount = membership rate \* month (if membership haven’t started)  Initial Amount = membership cost/day \* days left for current month + membership rate \* month left (if membership already started) |
| PAY NEW FEE |
| Initial Amount = rate \* quantity  Or  Initial Amount = custom rate \* quantity |

After a payment is stored into database, Payment and Payment Entry becomes Payment History and Payment Record, respectively.

Finance module Object Design

|  |  |
| --- | --- |
| GUI Layer | |
| PaymentEntryDisplay |  |
|  |  |
|  |  |
|  |  |