

## *Fairy* Dance Studio Time Sheet Management System



This assignment is a scaled down version of an actual final year project. The final year project team is working on a system to improve a dance studio school business operations in Singapore. The assignment is important to you because it aims to help you acquire the following learning outcomes and mind-sets:

- ① Gathering user requirements is a *continuous* process which concurrently requires rigorous prototyping and database design.
- ② Accurately determine the use case (how user uses the system) directly affects your solution's usability and *stabilizes* your database design.
- ③ Develop stronger server-side C# code patterns (especially working with date and time) which builds upon the fundamentals covered during lessons and practical exercises.
- ④ Develop stronger JavaScript client-side code fundamentals which is also based on the fundamentals covered during lessons.
- ⑤ Able to identify *possible mistakes* in database design at early stage which directly affects all the coding efforts and strategies at the business layer development and at the presentation layer development.

## Background Information



**Fairy** cultural arts and dance studio school has been using several cloud-based web application solutions which are hosted at servers in United States to manage their business operations for years. For example:

### 1 Using MINDBODY Online (Fig. 1) for Class enrolment system

The Fairy school management has purchased this web application solution service from MindBody Online which is currently based at United States. The MindBody Online system which allows their customers to purchase credits which can be used later to register for dance classes. When a customer signs up for a class session, the system deducts one credit from the customer's credit account (Appendix reference, Fig. A.1). The system has many useful functionalities which allow the school's staff to monitor the customers' class enrolment, perform analytics to analyse the customers' needs and much more. (**Appendix 7**, Fig. A.7 2)

#### MINDBODY Online - Everything Your Business Needs

[Ad] [software.mindbodyonline.com/Online](https://software.mindbodyonline.com/Online) ▼

Simplify Scheduling & Engage Clients With The #1 Business Management Software.

Get Real Time Updates · Since 2001 · Automate Your Daily Tasks · Scheduling & Payments

Services: Online scheduling, Payment processing, Automated tasks, Custom-tailored app, Point of sal..

Fig. 1

**2** The Fairy school management has been using another cloud solution which uses Google Sheets. Fairy school has many customers (e.g. Secondary School A, Secondary School B) which require instructors conduct lesson at **customer site premises**. The solution helps the instructors to clock their time in and time out after conducting lessons at the customer site. The school tracks these time sheet to bill the customers and pay the instructors.

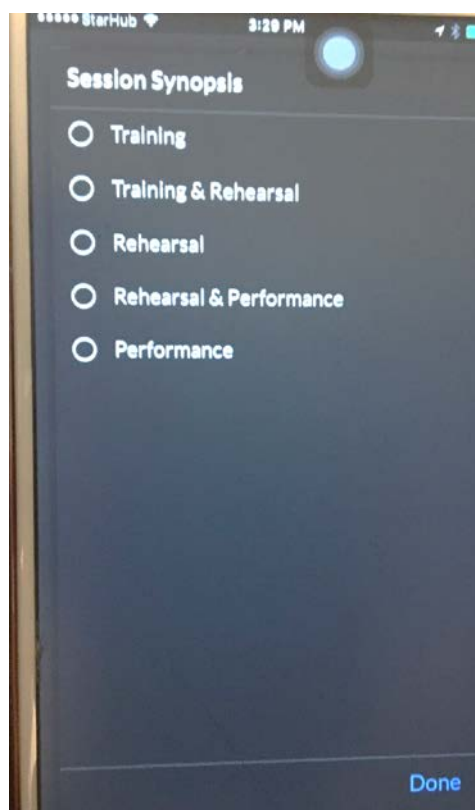
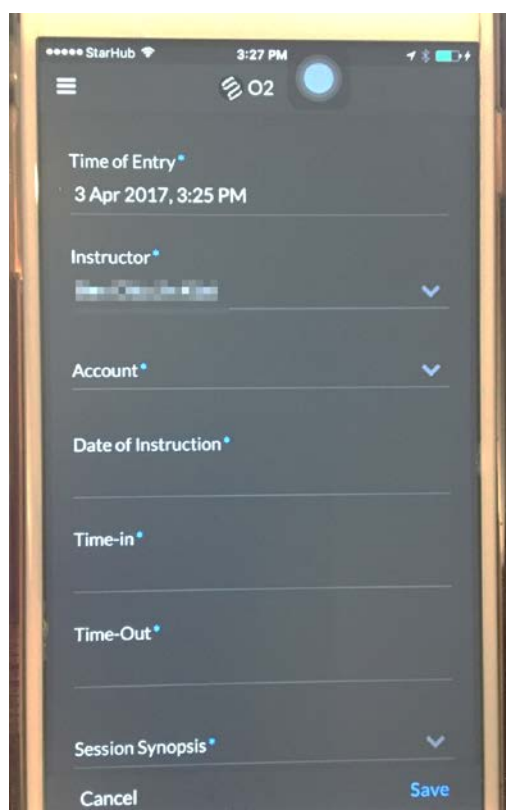


Fig. 2

## The Problem



Fig. 3

When business operations grow healthily year after year, the school management and dedicated staff are not comfortable to solely rely on cloud solutions. They often feel that, there are too much business operations data such as customer data, time sheet data, invoice data sitting inside the cloud solution. They did not take action to address this discomfort *until* an incident has happened.

MINDBODY Online **does not intend to support online payment system for users in Singapore**. The Fairy school's customers have to go through a bad user experience when comes to making a payment to purchase credits. They have to *queue*. This pain point has triggered or motivated the Fairy school management to engage Singapore Polytechnic final year project students to work with them on developing solutions which *can be hosted locally* in Singapore for them to rely upon in day to day business operations.

During the first FYP requirement gathering meeting, the final year project team was given a choice. Either choice **must lead to system implementation and actual user usage**.

**Choice 1**, work on the system which helps the school to:

- ① Manage time table of instructor lessons at customer site
- ② Generate billing invoice to site customer
- ③ Generate commission/payment invoice to instructor
- ④ Track instructor clock-in and clock-out (time in, time out) data when they conduct lessons at customer site.

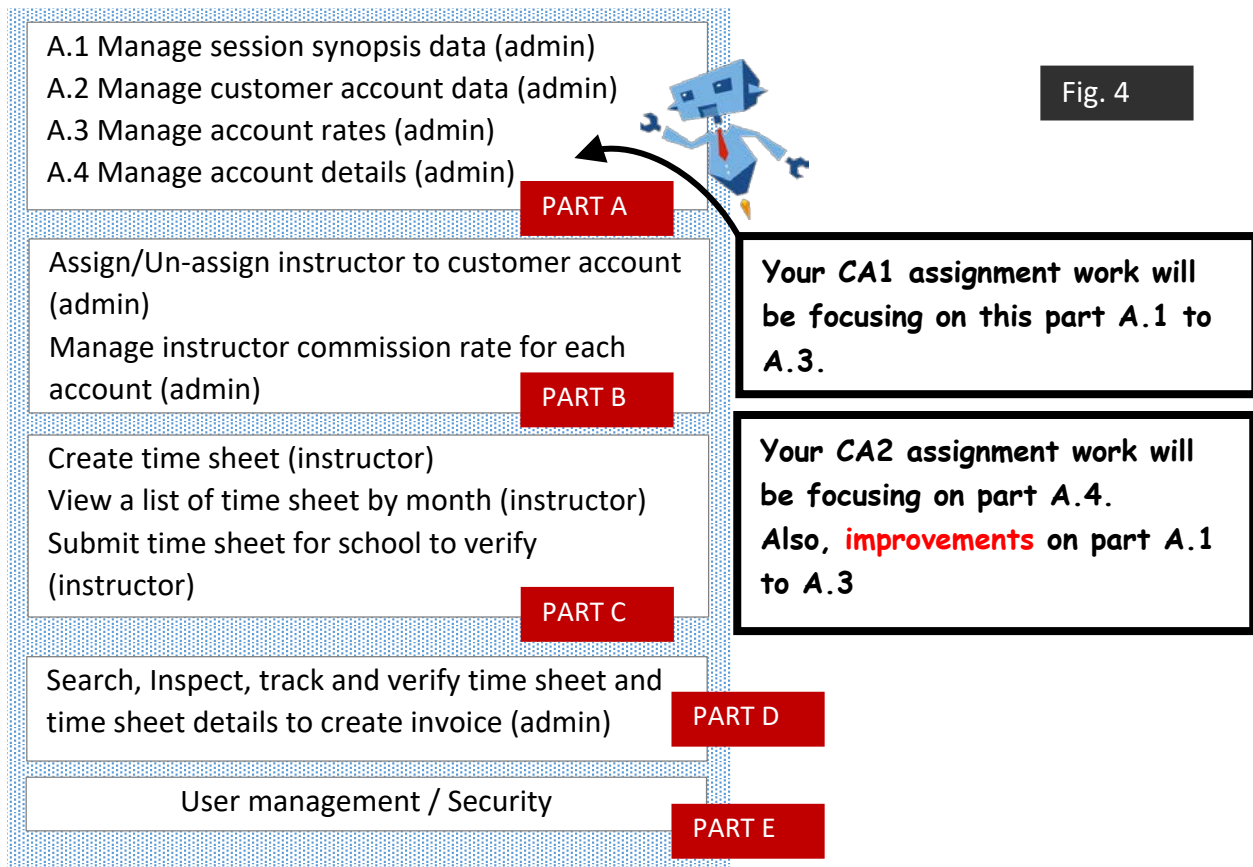


The final year project team chose **choice 1**.

**Choice 2**, work on a system to replace the functionalities which they often use in MINDBODY Online.

The figure below (Fig. 4) is a very high-level description on the system's building blocks, which can be useful to the Fairy school customer.

There are 5 main blocks. Fig. 4 also shows the parts you need for focus on.



### Assignment **CA 1** and **CA 2** System Requirements

There are three components which you need to work on in this Timesheet Management System. The three components are:

- ① Manage Session Synopsis (**CA1**)
- ② Manage Customer Account and Account Rate (**CA1**)
- ③ Manage Account Detail (Add Account Detail functionality - **CA2**, Update/Delete Account Detail – **CA2**)

**Use Case Name:** Create Session Synopsis user goal

**Identifier:** UC 01 [https://youtu.be/B\\_zLkfe8994](https://youtu.be/B_zLkfe8994)

**Description:**

An officer who holds an administrator account wants to create one session synopsis record which describes "Performance" inside the Timesheet Management System. Note that, session synopsis is their lingo in their business domain area. Session synopsis is something like our lesson type. For example, practical, tutorial and lecture. The administrator understands the consequences of not having this session synopsis record. For example, when an instructor begins to create a time sheet record to describe his time-in and time-out for a lesson which he has conducted at the customer site which involves performance, the instructor will have problem finding the "Performance" session synopsis to describe his lesson type.

**Pre-conditions:**

Not applicable

**Post-conditions:**

One session synopsis record is created in the database. The system is able to show the "Performance" session synopsis among other session synopsis description as options for the instructor to choose from, when he creates a timesheet.

**Basic Course of Action:**

1. The use case *begins* when officer (with administrator role) logon to the system, choose the "Manage session synopsis" navigation option to access the Create Session Synopsis view interface.
2. The administrator sees the interface and begin providing "Performance".
3. The administrator wants to let the instructor see this session synopsis information and choose it when the instructor fills up the necessary time-in time-out information during the process of timesheet creation. Therefore, the administrator selects a checkbox to indicate that this record is *visible* to all instructors.
4. The administrator submits the session synopsis data to the system.
5. The system receives the data and creates a new session synopsis record inside the database.
6. The system tracks and logs the user who has created the session synopsis record.
7. The use case ends *when* the administrator sees the notification message.

**Alternate Course A:** The administrator cannot send data to the system

A.1. Client-side validation logic detects the following:

- ① The user did not enter any text or "spaces only" for the session synopsis name.
- ② The administrator has accidentally entered an illegal name such as "**?Performance**" inside the text input for the session synopsis name while eating a burger.

A.2. The administrator sees some *helpful messages* and make changes accordingly before submitting again. The client-side JavaScript logic verifies that the information is complete, continues to collect the data and finally sends the collected data to the system. Note also, the administrator can also decide not to submit the data because she notices that she has created this session synopsis record before.

A.3. The use case ends.

**Alternate Course B:** The system rejects the new data entry

B.1. Server-side logic rejects the data entry when it detects that, the session synopsis name "Performance" conflicts with another session synopsis record which has the *same* session synopsis name.

B.2. The administrator sees some helpful messages and make changes accordingly before submitting again. Client-side logic verifies that the information is complete, continues to collect the data and finally sends the collected data to the system.

B.3. The use case ends.

---

---

**Use Case Name:** Update Session Synopsis user goal

**Identifier:** UC 02

**Description:**

An officer who holds administrator role account wants to update one session synopsis record which describes "**Performnce**" inside the Timesheet Management System. There was a typo in the respective session synopsis name.

**Pre-conditions:**

There is existing session synopsis record "**Performnce**" inside the system's database.

**Post-conditions:**

One session synopsis record is updated in the database. The system is able to reflect the new changes to the instructors and to the administrator.



**Basic Course of Action: (Reference: Appendix A.2 - Update Session Synopsis Use Case Flow)**

1. The use case *begins* when officer (with administrator role) logon to the system, choose the "Update session synopsis" navigation option to view the session synopsis information within the **Update Session Synopsis view interface**.
2. The administrator sees session synopsis name, "**Performmnce**" within the view interface's input element and begin changing the value to "**Performance**".
3. The administrator wants to let the instructor see this session synopsis information and choose it when the instructor creates the time sheet. Therefore, the administrator left the toggle switch interface alone so that this record remains *visible* to all instructors.
4. The administrator submits the session synopsis data to the system.
5. The system receives the data and creates a new session synopsis record inside the database.
6. The system tracks and logs the user who has created the session synopsis record.
7. The use case ends *when* the administrator sees the notification message.

**Alternate Course A:** The administrator cannot send data to the system

- A.1. Client-side validation logic detects the following:
  - ① The user did not enter any text or "spaces only" for the session synopsis name.
  - ② The administrator has accidentally entered an illegal name such as "**?Performance**" inside the text input for the session synopsis name while eating a Big Mac.
- A.2. The administrator sees some *helpful messages* and make changes accordingly before submitting again. Client-side logic checks whether the information is valid, then continues to collect the data and finally sends the collected data to the system.
- A.3. The use case ends when administrator sees a notification message.

**Alternate Course B:** The system rejects the new data entry

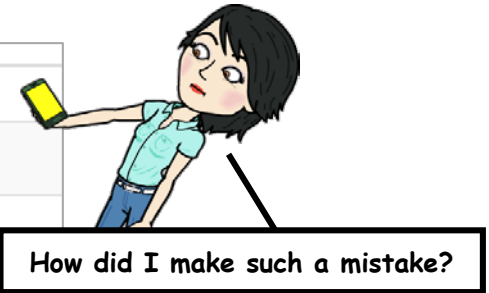
- B.1. Server-side logic rejects the data entry when it detects that the the session synopsis name "Performance" conflicts with another synopsis session record which has the *same* session synopsis name.
- B.2. The administrator sees some helpful messages and make changes accordingly before submitting. The client-side logic validates the information and sends the collected data to the system. The server-side logic checks whether the session synopsis name is unique and finally commit the changes inside the **SessionSynopsis** table.
- B.3. The use case ends when the administrator sees a notification message.

**Use Case Name:** Delete Session Synopsis user goal

**Identifier:** UC 03 Version 12 May 2017

**Description:** (video reference: [https://youtu.be/B\\_zLkfe8994](https://youtu.be/B_zLkfe8994))

A situation has occurred. There are two session synopsis records having "Training and Rehearsal" and "Training & Rehearsal". The administrator noticed it when an instructor gave her a call. There are two choices. The first choice is quickly set one of the session synopsis record as *invisible*. The second choice is delete one of them. The administrator **decides to delete the session synopsis which describes "Training and Rehearsal"**.



#	Actions	Session Synopsis Name	Visibility
1	<input type="checkbox"/>	Training	✓
2	<input type="checkbox"/>	Training & Rehearsal	✓
3	<input type="checkbox"/>	Rehearsal	✓
4	<input type="checkbox"/>	Rehearsal & performance	✓
5	<input type="checkbox"/>	Performance	✓
6	<input type="checkbox"/>	Training and Rehearsal	✓

**Pre-conditions:**

There is existing session synopsis record "**Training and Rehearsal**" inside the system's database.

**Post-conditions:**

One session synopsis record is permanently deleted (removed). The record is *not recoverable*. Users will never see this record again unless the administrator creates a new "Training and Rehearsal" record by accident or on purpose again.

**Basic Course of Action:**

1. The use case *begins* when the officer (with administrator role) logon to the system, finds the session synopsis record information inside the Manage Session Synopsis view interface, then clicks the "Update " button interface which corresponds to the row displaying the "**Training and Rehearsal**".
2. The system brings the administrator to the **Update Session Synopsis view interface**. The client-side logic will work closely with the server-side to display the correct session synopsis record details in the form input interface.
4. The administrator clicks the Delete button interface.



5. The administrator sees the modal dialog interface appearing asking her to confirm her delete action.
6. The administrator continues by clicking the Delete button interface within the modal dialog interface.
7. The system deletes the session synopsis record which describes "**Training and Rehearsal**". The system continues to bring the administrator back to the Manage Session Synopsis view interface. The administrator sees that the "**Training and Rehearsal**" session synopsis does not appear in the list.

**Alternate Course A:** The administrator clicks the Cancel button interface inside the modal dialog window.

- A.1. The administrator decides to abandon her delete action. The confirm delete modal dialog interface goes away, and the administrator can see the "**Training and Rehearsal**" session synopsis record details inside the Update Session Synopsis view interface.

**Alternate Course B:** The system *warns* the administrator that the session synopsis record has been used by some timesheet detail record. The instructors might have chosen this session synopsis "**Training and Rehearsal**" when they create their timesheet details. Technical note: Behind the scene, *there won't be any impact* to the existing timesheet records because the session synopsis name is *copied* into the timesheet detail record when the instructors create them.

- B.1. Additional sentences are seen within the confirm delete modal dialog interface. The information is enough to let the administrator decide whether to use the Manage Timesheet view interface (not within the scope of this assignment) to inspect these timesheet records.
- B.2. The administrator abandons the delete action.
- B.3. The use case ends when the confirm delete modal dialog interface vanishes.

**Use Case Name:** Create Customer Account user goal

**Identifier:** **UC 07** Version 12 May 2017

**Description:** Sometime in **March** 2017, the Fairy school's business team has signed an agreement with a customer, Secondary School E.

Fairy school management will assign **two** instructors to train students at Secondary School E during extra-curricular activities session. The customer has agreed on 100 SGD for the rate per hour charge. The customer is aware that Fairy school will send *two* invoices at the beginning of each month. Each bill describes the total charges for the services rendered by one instructor.

Fairy school officer need to create a new customer account record inside the system *first, before* she can continue assign two instructors to the customer account. *Without* these information, the system cannot help the two instructors to select to *correct* customer account to create time sheet data in the future to describe their clock-in and clock-out time.

**Pre-conditions:**

Not applicable

**Post-conditions:**

One *customer account* record is created in the database. At the same time, one *account rate* record is created inside the database.

In the future, if the administrator wants to assign two instructors to teach in SECONDARY SCHOOL E, the system can use the customer account record to help the school officer to further create two records which can describe two instructors associated to this customer account, Secondary School E.

In the future, if the administrator logon to the system to view/create/verify monthly invoice when she wants to issue a billing invoice to SCHOOL E, the system can use the *correct* account rate record information to perform the calculations.

**Basic Course of Action:** (Reference: Appendix A.3)

1. The use case *begins* when officer (with administrator role) wants to create a new customer account record.
2. The officer logon to the Timesheet Management system and chose the navigation option which is labelled as "Manage customer account". Video reference:  
<https://youtu.be/ynHngXuO4dw>
3. The system uses the correct view interface to display a list of existing customer accounts to the officer.
4. The officer clicks the "Add customer account" button interface to navigate to the Create Customer Account view interface.
5. The system displays the necessary data entry interface to let the officer describe two information. **(1)** The new customer, Secondary School E. **(2)** The rate per hour information which the officer wants the system to use when performing invoice billing calculations.

6. The officer indicates the following and click Save.

Account name	Effective start date	Effective end date	Rate per hour	Comments	IsVisible
SECONDARY SCHOOL E	1/5/2017	Empty	100 SGD per hour	Two lessons will be conducted (MON & FRI) each week. Both lessons are to be conducted from 7 PM to 9 PM.	True

When the system receives the data, the system notices that, the effective end date is not indicated. The system should treat this customer account's rate per hour information can only be used *after* 1st May 2017. Also, the system will use the customer account record's rate per hour information *forever* when comes to generating billing invoice to customer because the system treats an "empty" effective end date value as "open date".

When the system sees that the new customer account data has visibility set to true, the system understands that it needs to let the assigned instructor to see this customer account name "SECONDARY SCHOOL E" and choose it to begin creating timesheet data.

Account name	Comments	IsVisible
SECONDARY SCHOOL E	Two lessons will be conducted (MON & FRI) each week. Both lessons are to be conducted from 7 PM to 9 PM.	True

Rate per hour	Rate Effective start date	Rate Effective end date
100 SGD per hour	1/5/2017	Empty

When the system receives the data, it will check to see if the given account name *conflicts* with any existing customer account records' account name inside the database.

Finally, the system creates a new customer account record and one rate per hour (account rate) record (in **two separate** database tables) inside the database.

7. The system informs the officer that she has saved the customer account information.

8. The use case ends *when* the administrator sees the notification message.

**Alternate Course A:** The administrator is **stopped** by client-side validation logic to send data to the system's server-side for further processing.

A.1. Client-side validation logic detects that the administrator didn't provide any of the following:

① Account name ② Rate effective start date ③ Rate per hour

A.2. Client-side validation logic detects that the Rate effective end date is *earlier* than the Rate effective start date.

A.3. Client-side validation logic detects that the rate per hour value is *not numeric*.

A.4. The administrator sees some helpful messages, make changes to her input and finally resubmits the data. Client-side logic detects that the data is valid, collects the data and sends the data to the system.

A.5. The use case ends *when* the administrator sees the notification message.

**Use Case Name:** Create new Rate per Hour record for an existing customer account record user goal

**Identifier:** UC 08 Version 13 May 2017

**Description:**

Assuming that the system already has:

- ① One customer account record, **SECONDARY SCHOOL E**
- ② One account rate record which describes that SECONDARY SCHOOL E is charged 100 SGD for each hour. For example, if the assigned instructor has rendered 40 hour services to SECONDARY SCHOOL E for June 2017, the total invoice billing to that school will be 4000 SGD. The account rate record also describes that this rate is effective starting from 1st May 2017.

Account name	Comments	IsVisible
SECONDARY SCHOOL E	Two lessons will be conducted (MON & FRI) each week. Both lessons are to be conducted from 7 PM to 9 PM.	True

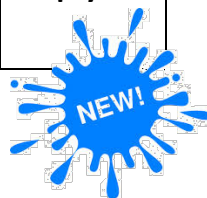
This is called the customer account record.

Rate per hour	Rate Effective start date	Rate Effective end date
100 SGD per hour	1/5/2017	Empty

This is called the account rate record.

The administrator needs to create an *additional account rate* record inside the system (see below) to describe the new rate per hour charge agreement.

Rate per hour	Rate Effective start date	Rate Effective end date
100 SGD per hour	1/5/2018	Empty



A new account rate record is required. This new account rate record must be able to describe that the customer SECONDARY SCHOOL E is going to be charged 120 SGD *starting* from 1st May **2018**.

**Pre-conditions:**

Refer to the use case (UC 08) description on the pre-conditions. The pre-conditions are required for **preparing test cases**. Before the customer in final year project accepts the system and use the system, a test plan (consists of many test cases) are required and the customer (user) shall sit alongside with you to use the system and inspect the expected results. A lot of test data is required before a proper test case activity is done together with the customer. What you write in the pre-conditions provides clarity on how to prepare the test environment before the test starts.

**Post-conditions:**

One new *account rate* record is created inside the database. The new account rate record *should not have an* effective start date which is earlier than the previous account rate record's effective start date.

The new account rate record's parent record, which is the customer account record "SECONDARY SCHOOL E" data is changed too. Because, the system needs to update the *main* record, to describe who was the last person changed the customer account's child record. Note that, the **AccountRate** table will not have any **UpdatedById** and **CreatedById** fields.

**Possible impact analysis**

If the new account record is having effective start date which is *earlier* than any existing account rate record, the system should generate a warning to the administrator on the *irregular data entry*.

**Basic Course of Action:** (Video reference: <https://youtu.be/F5VjeF1jUTg>)

1. The use case *begins* when an officer in Fairy School (who has user account with administrator role) wants to create a new customer account record.
2. The administrator logon to the Timesheet Management system and chose the navigation option which is labelled as "Manage customer account"
3. The system uses the Manage Customer Account view interface to display a list of existing customer accounts to the administrator.
4. The administrator clicks the "Manage Rate/Hour" button interface which corresponds to the row which displays the customer account "SECONDARY SCHOOL E" to navigate to the Manage Rate view interface.
5. The Manage Rate view interface displays one account rate information to the administrator.
6. The administrator finds and click the Add Rate button interface. The system brings the administrator to the Add Rate view interface.
7. The administrator provides the following data and submits.

Rate per hour	Rate Effective start date	Rate Effective end date
100 SGD per hour	1/5/2018	Empty

8. When the system receives the data and the system knows that this data is tied to the customer account "SECONDARY SCHOOL E". The system creates a new record inside the **AccountRate** table.



9. After creating the account rate record, the system informs the administrator that she has successfully saved the new account rate information. At the same time, the system checks whether the new account rate record's effective start date is earlier than other existing account rate records' effective start date. If there is, the system will warn the administrator.
10. When the administrator uses the Manage Rate view interface to view a list of account rate information for SECONDARY SCHOOL E, she should see **two** account rate information.
11. The use case ends.

**Alternate Course A:** The administrator is stopped by the client-side validation logic from sending her data to the system

- A.1. Client-side validation logic detects that the user did not enter any of the following:  
① Rate effective start date ② Rate per hour (Video reference <https://youtu.be/xyQRYSSzII0>)
- A.2. Client-side validation logic detects that the Rate effective end date is *earlier* than the Rate effective start date.
- A.3. Client-side validation logic detects that the rate per hour value is *not numeric*.
- A.4. The officer sees some helpful messages, make changes accordingly and submits the data again. Client-side logic checks the give input, collects the data and sends the data to the system.
- A.5. The use case ends *when* the administrator sees the notification message.

Create Account Rate for SECONDARY SCHOOL E

**Rate per hour**   
Please enter rate per hour

**Effective start date**    
Please enter start date

**Effective end date**

**Use Case Name:** Administrator update existing Rate per Hour record for an existing customer account record user goal

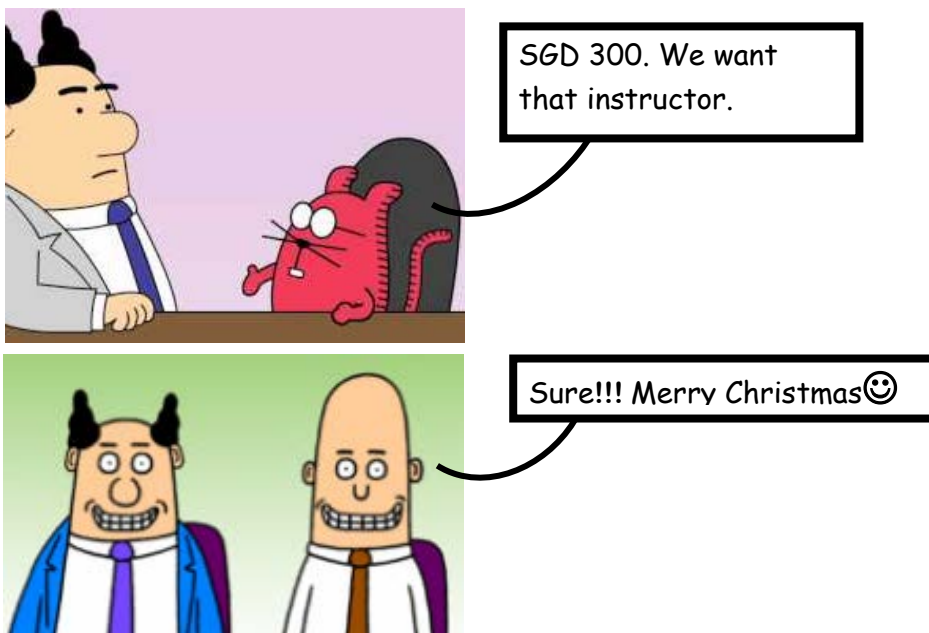
**Identifier:** **UC 09** Version 13 May 2017

**Description:**

Assuming that the system already has:

- ① One customer account record, **SECONDARY SCHOOL E**
- ② Two account rate records as described below:

Rate per hour	Rate Effective start date	Rate Effective end date
100 SGD per hour	1/5/2017	Empty
120 SGD per hour	1/5/2018	Empty



Sometime in November 2017, the customer, SECONDARY SCHOOL E's officer-in-charge came to see the Fairy School's boss personally. He shared to the Fairy School boss that, the instructor A's service is too excellent. The customer explained that, he received news that the instructor A will be re-assigned to give lessons at another customer site which happens to be SECONDARY SCHOOL E's competitor. After serious meeting on this matter, the customer is willing to offer SGD 300 per hour fee starting from 1st December 2017.

The Fairy School's boss happily tells the administrator to key into the system to "upgrade" the hourly rate charge for SECONDARY SCHOOL E record.

The officer-in-charge in Fairy School is having her overseas holiday. Therefore, the Fairy School's boss who also has an administrative account has to do it himself. He has two choices.

- ① Create a new account rate record which will be effective from **1st June 2017** for the SECONDARY SCHOOL E main customer account record. Delete the other account rate record which is effective from **1st May 2018**.
- ② Choose the *second* account rate record which is effective from **1st May 2018** and change it.

The Fairy School's boss decides to **update** the second account rate record. At the same time, he will edit the main customer account record's comment field to provide some information on the *reasons* for the change.

### Post-Conditions:

The system (behind the scene) will be update the main customer account record to reflect that, the Fairy School's boss is the most recent person who made changes to SECONDARY SCHOOL E record and child records. When the other officer comes back from her holiday, she will notice the changes and who has changed it.

Rate per hour	Rate Effective start date	Rate Effective end date
100 SGD per hour	1/5/2017	Empty
300 SGD per hour	1/12/2017	Empty

**Basic Course of Action:** (Video reference <https://youtu.be/zl0K2KlnNtQ>)

1. The use case *begins* when the administrator logon to the system and access the Manage Customer Account view interface.
2. The officer logon to the Timesheet Management system and chose the navigation option which is labelled as "Manage customer account"
3. The system uses the Manage Customer Accounts view interface to display a list of existing customer accounts to the administrator.
4. The administrator clicks the "Manage Rate/Hour" button interface which corresponds to the row which displays the customer account "SECONDARY SCHOOL E" to navigate to the Manage Rate view interface.
5. The Manage Rate view interface displays two account rate information to the administrator.
6. The administrator finds and click the Update button interface which corresponds to the row that displays an account rate record showing 120 SGD effective from 1st May 2018. The system brings the administrator to the Update Account Rate view interface.
7. The administrator provides the following data and submits.

Rate per hour	Rate Effective start date	Rate Effective end date
100 SGD per hour	1/5/2018	Empty

8. When the system receives the data and the system knows that this data is tied to the customer account "SECONDARY SCHOOL E". The system updates the correct account rate record inside the **AccountRate** table. At the same time, the system also updates the main parent customer account record to describe the user who has made the changes and the date-time of that action. Important supplementary video reference:  
<https://youtu.be/44O40LtbU8g>
9. After updating the account rate record, the system informs the Fairy School's boss that he has successfully updated the account rate information. At the same time, the system checks whether the updated account rate record's effective start date is earlier than any other existing account rate records' effective start date. If there is, the system will warn the administrator.
10. When the administrator uses the Manage Rate view interface to view a list of account rate information for SECONDARY SCHOOL E, he should see the updated account rate information.
11. The use case ends.

---

**Use Case Name:** **Administrator** update Customer Account general information user goal

**Identifier:** **UC 10** Version 16 May 2017



Refer to Appendix A.6 for details about this use case.

**Use Case Name:** **Administrator** create two account detail for SECONDARY SCHOOL E user goal

**Identifier:** **UC 12** Version 16 May 2017

**Description:**

Assuming that the system already has:

- ① One customer account record, **SECONDARY SCHOOL E**
- ② Two account rate records as described below:

Rate per hour	Rate Effective start date	Rate Effective end date
<b>100</b> SGD per hour	1/5/2017	<b>Empty</b>
<b>120</b> SGD per hour	1/15/2018	<b>Empty</b>

Sometime **in April 2017**, the customer SECONDARY SCHOOL E has *agreed* that weekly lessons will be conducted on every Monday and Friday. On each Monday, lesson starts at 7 PM and ends at 9 PM. On each Friday, lesson starts at 7 AM and ends at 9 AM too. The administrator must provide this lesson time table information into the Timesheet Management System *before* May 2017.

Without these information, the system is unable to use them to calculate out all the possible dates for the instructor to choose when creating a time sheet. For example:

When the instructor uses the system to create a time sheet to clock his time in and time out in May 2017, the system can calculate out the following dates for the instructor to choose from.

DATE
2017-05-01 00:00:00.000
2017-05-08 00:00:00.000
2017-05-15 00:00:00.000
2017-05-22 00:00:00.000
2017-05-29 00:00:00.000

When the instructor uses the system to create a time sheet to clock his time in and time out in December 2017, the system can calculate the following dates for the instructor to choose from.

DATE
2017-12-04 00:00:00.000
2017-12-11 00:00:00.000
2017-12-18 00:00:00.000
2017-12-25 00:00:00.000

Important note: **If the instructor tries to create a time sheet data at any time in April 2017, the system will not display any dates for the instructor to choose.**

The officer in charge who has the administrative account can use the Create Account Detail view interface to achieve the user goal.

### Post-Conditions:

The system (behind the scene) will be update the main customer account record about the user who has made changes to SECONDARY SCHOOL E record and child records and the date-time of the action taken. The system creates **two** account detail record information within the **AccountDetail** table. For detailed description, refer to the Appendix A.4 (Fig. A4.7).

### Basic Course of Action:

1. The use case *begins* when the administrator logon to the system and access the Manage Customer Account view interface.
2. The administrator logon to the Timesheet Management system and chose the navigation option which is labelled as "Manage customer account"
3. The system uses the Manage Customer Accounts view interface to display a list of existing customer accounts to the administrator.
4. The administrator clicks the "Manage Account Details" button interface which corresponds to the row which displays the customer account "SECONDARY SCHOOL E". The system directs the user to Manage Account Details view interface.
5. The Manage Account view interface tells the administrator that, "there are no account detail data for SECONDARY SCHOOL E". The administrator finds the Add Account Detail button interface and clicks it.
6. The administrator sees a form interface within the Add Account Detail view interface (Appendix A.4 - Fig. A4.4).
7. The administrator provides the following data and submits.

Week Day Name	Start time	End time	Effective from	IsVisible
Monday	07:00 PM	09:00 PM	01/05/2017	True

8. When the system receives the data and the system knows that this data is tied to the customer account "SECONDARY SCHOOL E". The system creates one account detail record inside the **AccountDetail** table. At the same time, the system also updates the main parent customer account record to describe the user who has made the changes and the date-time of that action.
9. After creating the account detail record, the system informs the administrator that he has successfully created one account detail information.
11. Within the Add Account Detail view interface, the administrator continues to provide the second piece of information to the form input.



Week Day Name	Start time	End time	Effective from	IsVisible
Friday	07:00 PM	09:00 PM	01/05/2017	True

12. When the system receives the data and the system knows that this data is tied to the customer account "SECONDARY SCHOOL E". The system creates the second account detail record inside the **AccountDetail** table. At the same time, the system also updates the main parent customer account record to describe the user who has made the changes and the date-time of that action.
13. The use case ends.

## Appendix A.1 - Timesheet

**Use Case Name:** Instructor creates timesheet data to clock his service hours' user goal

**Identifier:** UC 14 Version 14 May 2017

### WEBA Assignment Note:

The scope of CA1 and CA2 assignment **does not involve** developing the timesheet functionality and the invoice management functionality. This section shares how an instructor use the timesheet functionality. The documentation for UC14 is **absolutely not detailed** enough. **A detailed use case documentation can be used to prepare unit test plans which are required for end-user testing phase at the end of the Final Year Project.**

**This UC 14 is given as a reference here to allow WEBA learners appreciate the entire system's objective.**



### Use Case Description:

**Who:** The instructor.

**When:** When the instructor has finished giving his lesson.

**Where:** The lesson is done at the customer's site. (Not at Fairy School premises)

**How:** Access his mobile web browser and logon as a user to find the Create Timesheet function.

The following three figures below and next page describe the mobile solution which Fairy school users need.

Fig. UC 14.1

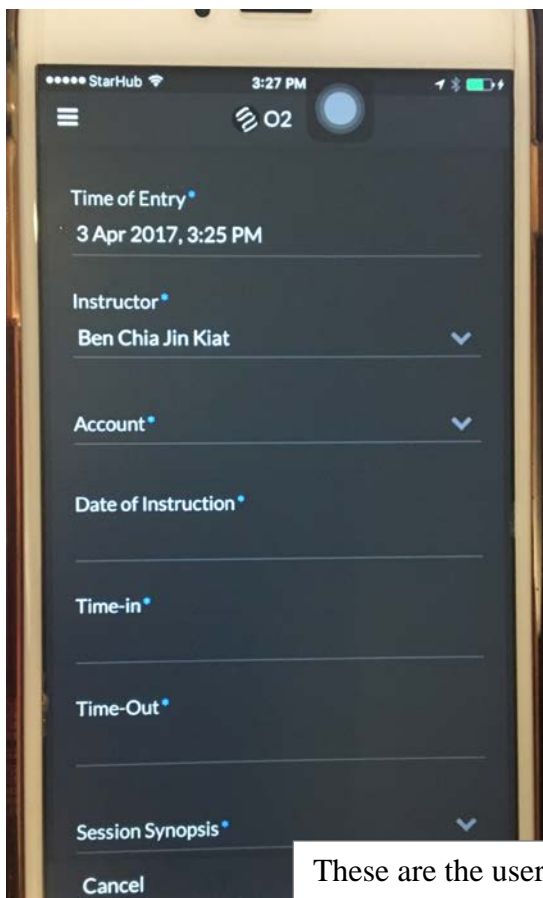
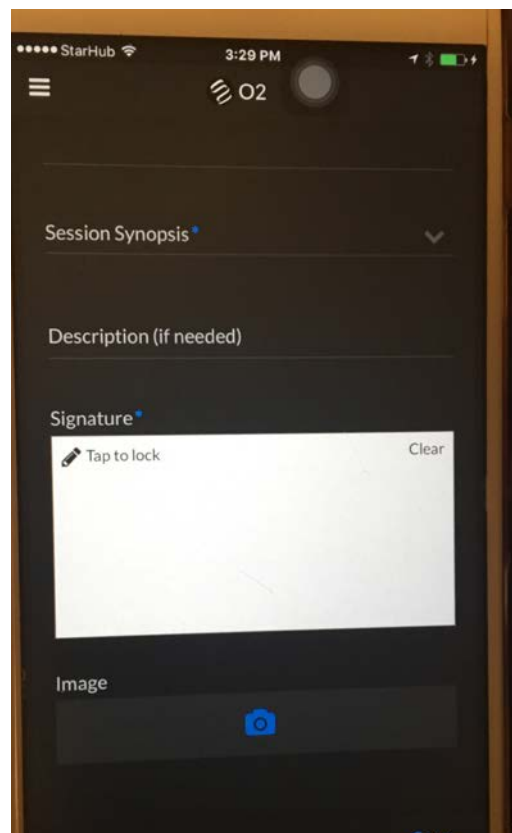


Fig. UC 14.2



These are the user interfaces which instructor role user uses to achieve his user goal in creating a timesheet and timesheet detail record in the system.

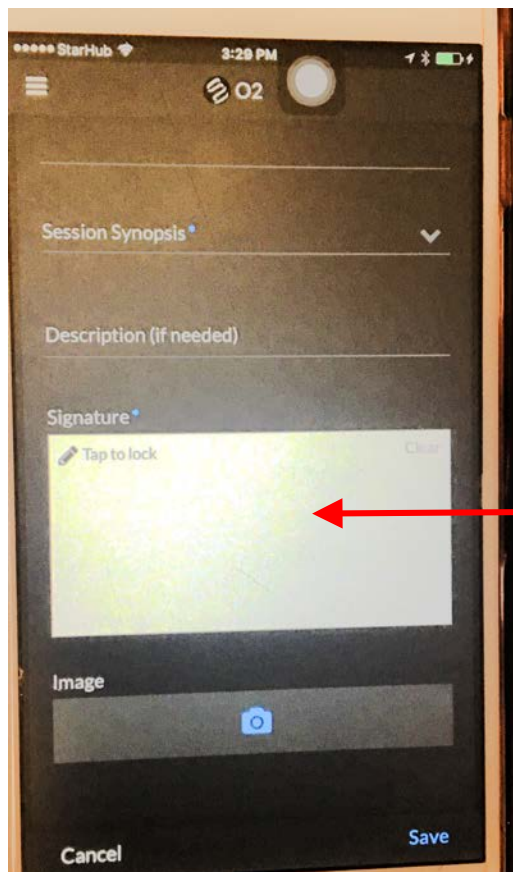


Fig. UC 14.3

The instructor user does not provide a digital signature here. After conducting a lesson, the instructor has to find the officer in charge at the customer site (e.g. SECONDARY SCHOOL E) to sign it.

The Fairy school management will ignore timesheet detail records which do not have digital signature.

### Pre-conditions:

The system database must have all the necessary supporting data in the respective database tables so that the instructor can achieve his user goal (create a timesheet record *after* giving a lesson). The supporting data is:

- ① Session synopsis records (records inside the **SessionSynopsis** table)
- ② Customer account record (records inside the **CustomerAccount** table)
- ③ The record which links the respective instructor to the customer account (records inside the many-to-many join table, **InstructorAccount**)
- ④ The Account detail record (the record which describes the lesson time table at the customer site)

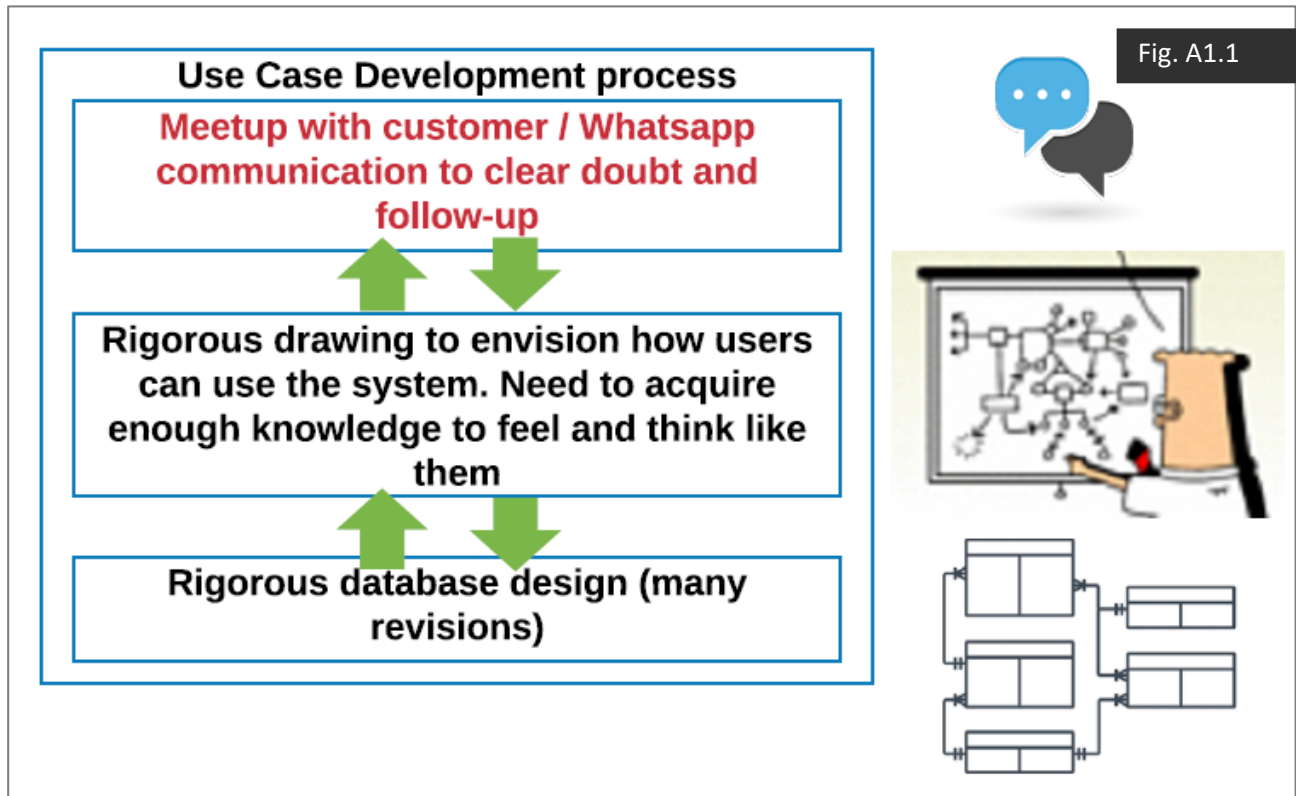
### Post-conditions:

Affected tables: **Timesheet**, **TimesheetDetail**, **TimesheetDetailSignature**

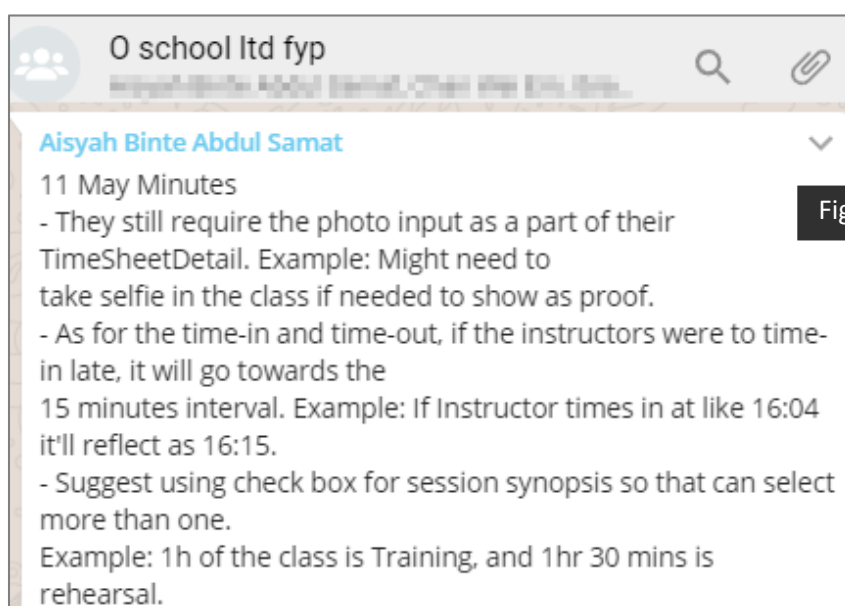
If the instructor uses the system to create his first timesheet record for June 2018, the system will create one timesheet main record inside the Timesheet parent table, **Timesheet** and one child timesheet detail record inside the **TimesheetDetails** table.

## What did the final year project team do, to prepare the use case detailed description for **Timesheet**?

The process is a repetitive cycle *until* the final year project team is able to think like the customer to validate the usefulness of the system. There is a deadline for this process which is maximum 3 weeks due to the Phase 1 documentation and presentation which happens at week 5 - week 6.



The chat figure below and next page, illustrates the weekly and daily communication with the customer.



- They would want a summary of all the TimeSheetDetails of current month before submitting the final TimeSheet. This is so they can double-check and confirm. If an Instructor realises that he has keyed in the wrong account detail within the TimeSheetDetail and attempts to make changes to it even after it has been signed and saved, he may not do so. He is only allowed to delete this TimeSheetDetail, and redo it (Mr Kenny says like need to do this in front of the admin so admin can make sure that it's legit). Example: InstructorA sees the summary of TimeSheetDetails, sees that for TimeSheetDetail1 he keyed in AccountDetailB instead of AccountDetailC. But he has already gotten the signature which is the required field for all TimeSheetDetails. InstructorA cannot edit to AccountDetailC or any other inputs bcz already has signature and saved after all. InstructorA can only delete TimeSheetDetail1, receive a re-prompt entry and recreate a new one with AccountDetailC. So there will be a new TimeSheetDetail2, even though it may be keyed on 31 May, the admin still should be able to know that the class was originally conducted on 10 May. They also want a time-of entry.

- The rate differs according to Instructor and Accounts. Rate does not usually halfway thru the month.

- Mr Kenny says that Instructors are able to sign all of the TimeSheetDetails at the end of the month, but he mentioned that it is a bad practice, he wants to discourage it.. If it happens, the admin might need to talk to the Instructor about it.

19:45

- A bonus would be that Instructor can see his past TimeSheets in past months. Another thing is that a notice like "Congratulations! So far you have earned \$\_\_\_ in \_\_\_ School!"

19:48

Fig. A1.2 Part 2

We have to go through the meeting minutes **many times** and at the same time, check the use case documentation/check the draft UI/research on database design and suitable technologies.



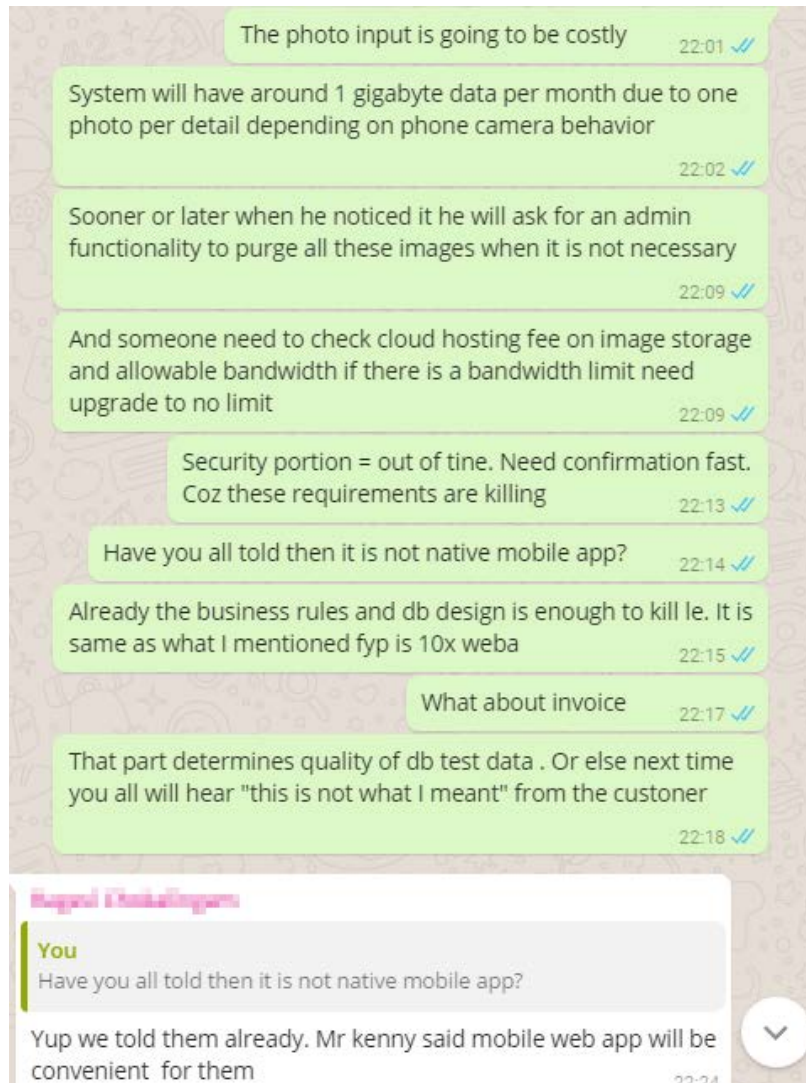


Fig. A1.2 Part 3

Teachers can guide learners to map out a plan and execute the plan.

Use case study is just the beginning. If did not do correctly, learners do not have the bandwidth to deal with cloud hosting, security, database sizing, concurrent users issue etc.





## Appendix A.2 - Update Session Synopsis Use Case Flow

Time Sheet Management    Manage User    Administration ▾

**Manage Session Synopses**

- Manage customer ccount
- Create customer ccount
- Manage session synopsis
- Create session synopsis

**Manage Session Synopses**

Add session synopsis

#	Actions	Session Synopsis Name	Visibility	Created by	Updated by	
1	▾	Training	✓	SUSAN	SUSAN	Update
2	▾	Training & Rehearsal	✓	SUSAN	SUSAN	Update
3	▾	Rehearsal	✓	SUSAN	SUSAN	Update
4	▾	Rehearsal & performance	✓	SUSAN	SUSAN	Update
5	▾	Performnce	✓	SUSAN	DANIEL	Update

5 ▾ Performance

Update session synopsis

**Update Session Synopsis Record**

Session synopsis name: Performance

Visibility: ☒ ON

Delete Save Cancel

**Update Session Synopsis Record**

Session synopsis name: Performance

Visibility: ☒ ON

Delete Save Cancel

**Appendix A.3 - Create Customer Account Use Case Flow**

State of the **CustomerAccount** table *before* user case ends.

Fig. A3.2 Part 1

ILOVECODE...merAccount × ILOVECODE\...ccountRate					
	CustomerAccountId	Account...	Comments	IsVisible	CreatedAt
	5	SCHOOL A	Weekly one lesson (Saturday)	True	2016-12-15 1
	6	SCHOOL B	Weekly two lessons (total 5 hours) - MON...	True	2017-11-20 1
	7	SCHOOL C	Weekly two lessons (total 4 hours) - FRI a...	True	2017-12-20 1
	8	SCHOOL D	Lessons (2 hour each lesson) to be condu...	True	2017-12-20 1

	IsVisible	CreatedAt	Created...	Update...
turday)	True	2016-12-15 15:4...	3	2016-12-... 3
otal 5 hours) - MON...	True	2017-11-20 15:4...	3	2017-11-... 3
otal 4 hours) - FRI a...	True	2017-12-20 15:4...	3	2017-05-... 3
esson) to be condu...	True	2017-12-20 15:4...	3	2017-05-... 3

Fig. A3.2 Part 2

State of the **AccountRate** table *before* the user case ends.

Fig. A3.3

AccountRateId	CustomerAccountId	RatePerHour	EffectiveStartDate	EffectiveEndDate
<i>NULL</i>	<i>NULL</i>	<i>NULL</i>	<i>NULL</i>	<i>NULL</i>

Refer to Fig. A3. 4 Part 1 and Fig. A3. 4 Part 2, to observe the state of the **CustomerAccount** table *after* user case ends. The new customer account record has a unique id value of **9**. Note that the database engine decides the value for the **CustomerAccountId** field.

Fig. A3.4 Part 1

CustomerAcc...	AccountName	IsVisible	Comments	UpdatedAt	Updated
5	SCHOOL A	False	Weekly one less...	2017-05-15 2...	2
6	SCHOOL B	True	Weekly two less...	2014-12-20 1...	2
7	SCHOOL C	True	Weekly two less...	2017-05-01 1...	2
8	SCHOOL D	True	Many lessons t...	2017-03-01 1...	2
9	SECONDARY SCHO...	True	Two lessons wil...	2017-03-01 1...	2

ments	UpdatedAt	UpdatedByld	CreatedAt	CreatedByld
one less...	2017-05-15 2...	2	2017-05-01 15:4...	2
two less...	2014-12-20 1...	2	2014-12-20 15:4...	2
two less...	2017-05-01 1...	2	2017-05-01 15:4...	2
essons t...	2017-03-01 1...	2	2017-03-01 15:4...	2
ssons wil...	2017-03-01 1...	2	2017-03-01 15:4...	2

Fig. A3.4 Part 2

At the same time, a child record which describes the account rate information is created. The record is generated inside the **AccountRate** table. The system ensures that the **CustomerAccountId** field value of the new **AccountRate** table record, has a value of **9** so that this record can be linked to the main parent record inside the **CustomerAccount** table.

Fig. A3.5

ILOVECODE\...bo.UserInfo ILOVECODE\...ccountRate × ILOVECODE...merAccount					
	AccountRateId	CustomerAccountId	RatePerHour	EffectiveStartDate	Effective
	1	9	100.00	2017-05-01 00:00:00.0000000	<i>NULL</i>

## Appendix A.4 Account Detail Database Design concept and Create Account Detail view interface behaviour

### Background information about the account detail term

We needed a table which can capture the weekly lesson time table information for each customer account. The following are two examples:

A customer wants to have two lessons conducted each week such as one lesson on Sunday (9 AM to 12 PM) and one lesson on Monday (10 AM to 12 PM).

Another customer such as SECONDARY SCHOOL E, wants a lesson to be conducted on Monday (7 PM to 9 PM) and another to be conducted on Friday each week (7 PM to 9 PM).

There was a difficulty in coming up with the right name for a database table which *captures* the *weekly time table* information for the respective customer account record (inside the

**CustomerAccount** table). We referenced the content at

[https://www.ntu.edu.sg/home/ehchua/programming/sql/relational\\_database\\_design.html](https://www.ntu.edu.sg/home/ehchua/programming/sql/relational_database_design.html)

which mentioned about order (parent table) and order detail (child table). We noticed that the child record which captures the details of weekly lesson time table information shares similarity with the order detail. Therefore, we have decided to use the **AccountDetail** table name. When the customers got used to our terms, they have started to apply that word as well.

Eventually, this "account detail" is used as a "term" during the conversation when final year project students work with the customers (Fairy School). Account detail means "*weekly lesson time table whereby the respective instructor is supposed to be at the customer's site to conduct lessons*".

Refer to Fig. A4.1 and Fig. A4.2 Part A and Part B. Study the records inside the **CustomerAccount** table and the **AccountDetail** table first.

ILOVECODE...merAccount		ILOVECODE\...countDetail		
	CustomerAccou...	AccountName	IsVisible	Comme...
	5	SCHOOL A	True	Weekly ...
	6	SCHOOL B	True	Weekly t...
	7	SCHOOL C	True	Weekly t...

Fig. A 4.1  
CustomerAccount

ILOVECODE\...countDetail		ILOVECODE\...countDetail			
	Account...	CustomerAccountId	DayOfWeekNumber	EffectiveStartDate	Effec
	1	6	2	2017-06-01 00:00:...	NULL
	3	6	1	2017-06-01 00:00:...	NULL

Fig. A 4.2 Part A  
AccountDetail

EffectiveStartD...	EffectiveEndDate	StartTimeInMinutes	EndTimeInMinutes	IsVisible
2017-06-01 00:00:...	NULL	600	720	True
2017-06-01 00:00:...	NULL	540	720	True

Fig. A 4.2 Part B  
AccountDetail

Observe the second record inside the **CustomerAccount** table. Within the **CustomerAccount** table, there is a customer account information "SCHOOL B" which can be uniquely identified by a value of **CustomerAccountId**, 6.

Study the two records inside the **AccountDetail** table. If the system can function correctly, the system should help the administrator to create two account detail information within this **AccountDetail** table.

The first record describes that, there is one lesson to be conducted on every Monday from 10 AM to 12 PM at "SCHOOL B". The field value for the **StartTimeInMinutes** is 600 (600 minutes from 12 midnight) and the field value for the **EndTimeInMinutes** is 720 (720 minutes from 12 midnight). This account detail record is *visible* (**IsVisible** field value is true). The system will use this account detail record to calculate an actual date for the instructor to choose when he is creating a time sheet data.

The second record inside the **AccountDetail** table is describing that, one lesson should be conducted at "SCHOOL B" on every Sunday from 9 AM to 12 PM. The value of 540 inside the **StartTimeInMinutes** field means 540 minutes from 12 midnight. The value of 720 in the **EndTimeInMinutes** field means 720 minutes from 12 midnight. This account detail record is also marked as visible due to the value of **true** inside the **IsVisible** field. This means that the system will use this account detail information to calculate the actual date for the customer to choose when he creates a time sheet data.

Fig. A 4.3 Part A  
AccountDetail

ILOVECODE\...countDetail × ILOVECODE\...countDetail					
	Account...	CustomerAccountId	DayOfWeekNumber	EffectiveStartDate	Effec
	1	6	2	2017-06-01 00:00:...	NULL
	3	6	1	2017-06-01 00:00:...	NULL

Fig. A 4.3 Part B AccountDetail

EffectiveStartD...	EffectiveEndDate	StartTimeInMinutes	EndTimeInMinutes	IsVisible
2017-06-01 00:00:...	NULL	600	720	True
2017-06-01 00:00:...	NULL	540	720	True

I have decided on using total minutes from midnight technique to store time value after checking out this discussion at:  
<http://stackoverflow.com/questions/538739/best-way-to-store-time-hhmm-in-a-database>

Fig. A4.5 is the first version user interface that the team has envisioned to assist the administrator to create one account detail information which describes that, one lesson should be conducted on Monday every week from 7:00 PM to 9:00 PM for SECONDARY SCHOOL E. The account detail information is visible so that the system can calculate out all the possible dates for a particular month. The instructor can choose the generated dates when he is creating a time sheet data.

### Create Account Detail for SECONDARY SCHOOL E

Week day name
Monday

Check current account details

Lesson start time
7:00 PM

Lesson end time
9:00 PM

Effective start date
01/05/2017

Effective end date

Is visible
ON

Save
Cancel

Day	Start Time	End Time	Effective Start Date	Effective End Date	Visible
<p><i>There is no account detail data for this customer account record at the moment.</i></p> <p><i>Instructors will not be able to create timesheet data without account detail information.</i></p>					

Fig. A 4.4

We drew this on an exercise book or papers to **envision** the likely view interface which helps user to create customer account. We did not have this user interface programmed yet when we were trying to figure out the database design.



The Fig. A4.5 Part 1 and Fig A4.5 Part 2 collectively describe the record content inside the **AccountDetail** table *after* the administrator has clicked the Save button interface.

Fig. A4.5 Part 1

AccountDetailId	CustomerAccountId	DayOfWeekNumber	EffectiveStartDate
8	9	2	2017-05-01 00:00:...

EffectiveStartDate	EffectiveEndDate	StartTimeInMinutes	EndTimeInMinutes	IsVisible
2017-05-01 00:00:...	NULL	1140	1260	True

Assume that SECONDARY SCHOOL E account record id is 9.

Fig. A4.5 Part 2



Fig. A4.6 describes the administrator creating the **second** account detail information which describes that, one lesson should be conducted on Friday every week from 7:00 PM to 9:00 PM for SECONDARY SCHOOL E. The account detail information is visible so that the system can calculate out all the possible dates for a particular month. The instructor can choose the dates when he is creating a time sheet data. Fig. A4.7 describes the outcome inside the **AccountDetail** table.

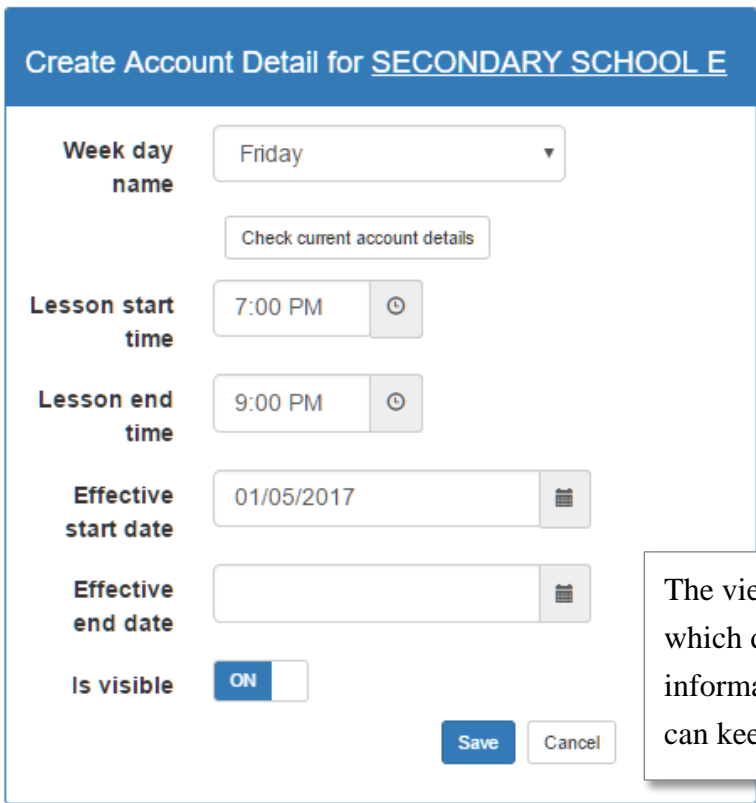


Fig. A4.6

The view interface has JavaScript logic which displays all the account detail information here so that the administrator can keep track what she has been adding.

Day	Start Time	End Time	Effective Start Date	Effective End Date	Visible
Friday	07:00 PM	09:00 PM	01/05/2017	NA	✓

Fig. A4.7

AccountDetailId	CustomerAccountId	DayOfWeekNumber	EffectiveStartDate
1	6	2	2017-06-01 00:00:...
3	6	1	2017-06-01 00:00:...
8	9	2	2017-05-01 00:00:...
9	9	6	2017-05-01 00:00:...

EffectiveStartDate	EffectiveEndDate	StartTimeInMinutes	EndTimeInMinutes	IsVisible
2017-06-01 00:00:...	NULL	600	720	True
2017-06-01 00:00:...	NULL	540	720	True
2017-05-01 00:00:...	NULL	1140	1260	True
2017-05-01 00:00:...	NULL	1140	1260	True

## Appendix A.5 Create Account Detail view interface **checks** for overlapping start time and end time on the **same day**

During the initial phase of the final year project, Fairy School final year project client is **not sure** whether the system should *restrict* the administrator from creating account detail information with overlapping start time and end time.

Refer to Fig. A5.1. The administrator uses the Create Account Detail view interface to define the third account detail information for SECONDARY SCHOOL E. The administrator is providing start time and end time which overlaps with the other account detail information which is from 7 PM to 9 PM on Monday each week.

### Create Account Detail for SECONDARY SCHOOL E

**Week day name** Monday ▼

Check current account details

**Lesson start time** 8:00 PM ⌚

**Lesson end time** 10:00 PM ⌚

**Effective start date** 01/05/2017 📅

**Effective end date**  📅

**Is visible** ON

Save
Cancel

Fig. A5.1

Day	Start Time	End Time	Effective Start Date	Effective End Date	Visible
Monday	07:00 PM	09:00 PM	01/05/2017	NA	✓
Friday	07:00 PM	09:00 PM	01/05/2017	NA	✓

Page 34 of 39

Refer to the figure below. When the administrator clicks the Save button interface, the system will check all the account detail records which are tied to the SECONDARY SCHOOL E main parent record, for **overlapping start and end time**. If there is any, the system will stop the administrator from creating the new account detail information.

Fig. A5.2

Warning: The data might conflict with

Day	Start time	End time	Effective start date	Effective end date	Visible
Monday	07:00 PM	09:00 PM	01/05/2017	n/a	✓

**Week day name**

Monday ▼

Check current account details

**Lesson start time**

8:00 PM

**Lesson end time**

10:00 PM

**Effective start date**

01/05/2017

**Effective end date**

**Is visible**

ON

Save

Cancel

Day	Start Time	End Time	Effective Start Date	Effective End Date	Visible
Monday	07:00 PM	09:00 PM	01/05/2017	NA	✓
Friday	07:00 PM	09:00 PM	01/05/2017	NA	✓

## Appendix A.6 Administrator update Customer Account information use case flow

The user (who has administrator account role) usually has two motivations to update the customer account information:

### Provide comments:

The first motivation is to allow other administrators aware of changes made to a particular customer account and the respective child records (account rate and account detail).

### Set visibility status:

The second motivation is due to under special circumstances such as the SECONDARY SCHOOL E's premise is under renovation and lessons need to be discontinued for one month.

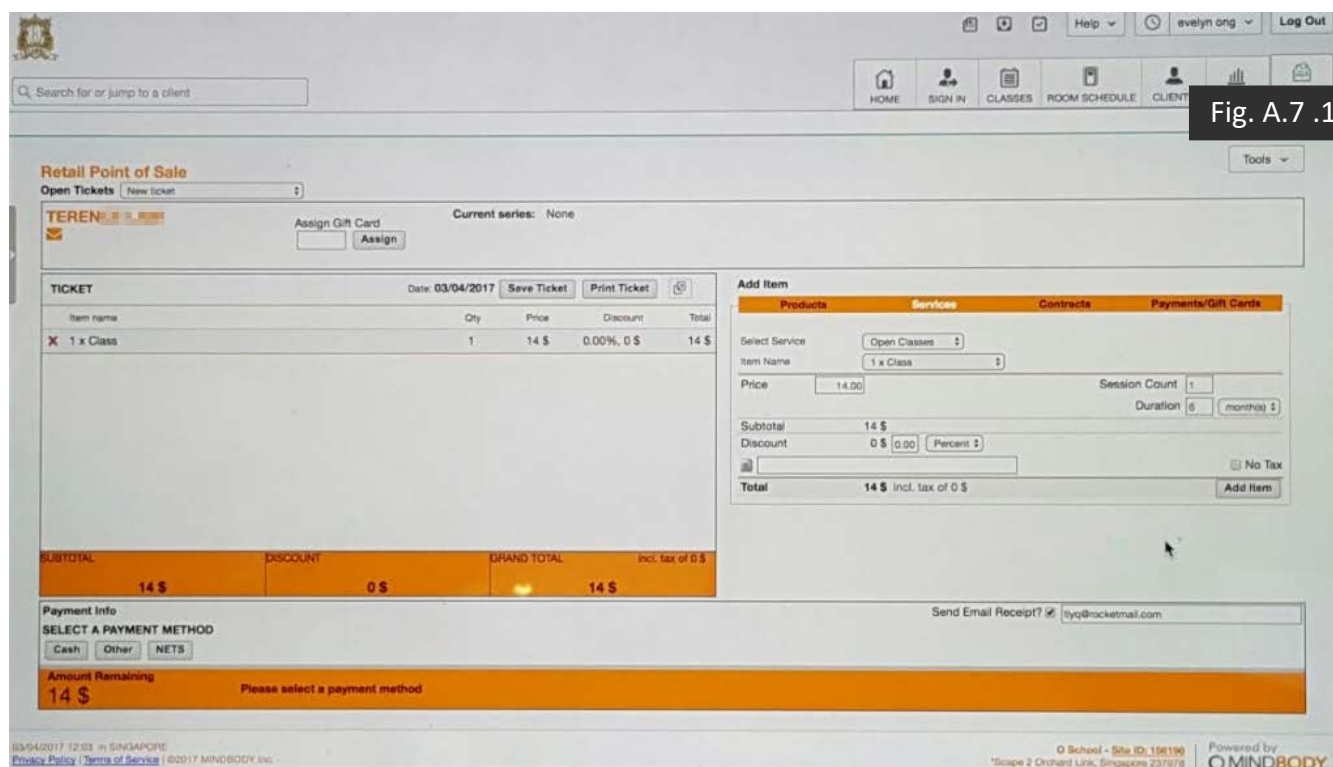
Timesheet Management System objective is not to capture detailed customer activity information nor provide a complex customer account management module. Timesheet Management system's goal is to help the Fairy School stakeholders efficiently capture time in and time out timesheet data from instructor to generate invoice payment. Therefore, a simple visibility setting is good enough to disallow the instructor user to see this SECONDARY SCHOOL E account name appearing in his view for one month.

The screenshot displays a web form titled "Update Customer Account". It contains three main input fields: "Account name" with the value "SECONDARY SCHOOL E", "Visibility" with a toggle switch set to "ON", and "Comments" with a text area containing the following text: "Two lessons will be conducted (MON & FRI) each week. Both lessons are to be conducted from 7 PM to 9 PM. Modified the 2nd account rate info from 120 SGD to 300 SGD effective from 1st Dec 2017. (By Kenny)". At the bottom right of the form are three buttons: "Delete" (red), "Save" (blue), and "Cancel" (white).

## Appendix A.7 Information on MINDBODY system which assists Fairy School in day to day business operations. (Although not related to assignment but it is important insights about FYP domain knowledge gathering process.

Fig. A.7 1 and Fig. A.7 2 MINDBODY Online Sample

A photo taken during the final year project user requirement gathering on how users use the MindBody Online to do their business operations.



**Retail Point of Sale**

Open Tickets:

Assign Gift Card:  Current series: None

**TICKET** Date: 03/04/2017

Item name	Qty	Price	Discount	Total
1 x Class	1	14 \$	0.00%, 0 \$	14 \$

**Add Item**

Select Service:

Price: 14.00 Session Count: 1 Duration: 6 months

Subtotal: 14 \$ Discount: 0 \$ 0.00 % Total: 14 \$ incl. tax of 0 \$

**Payment Info**

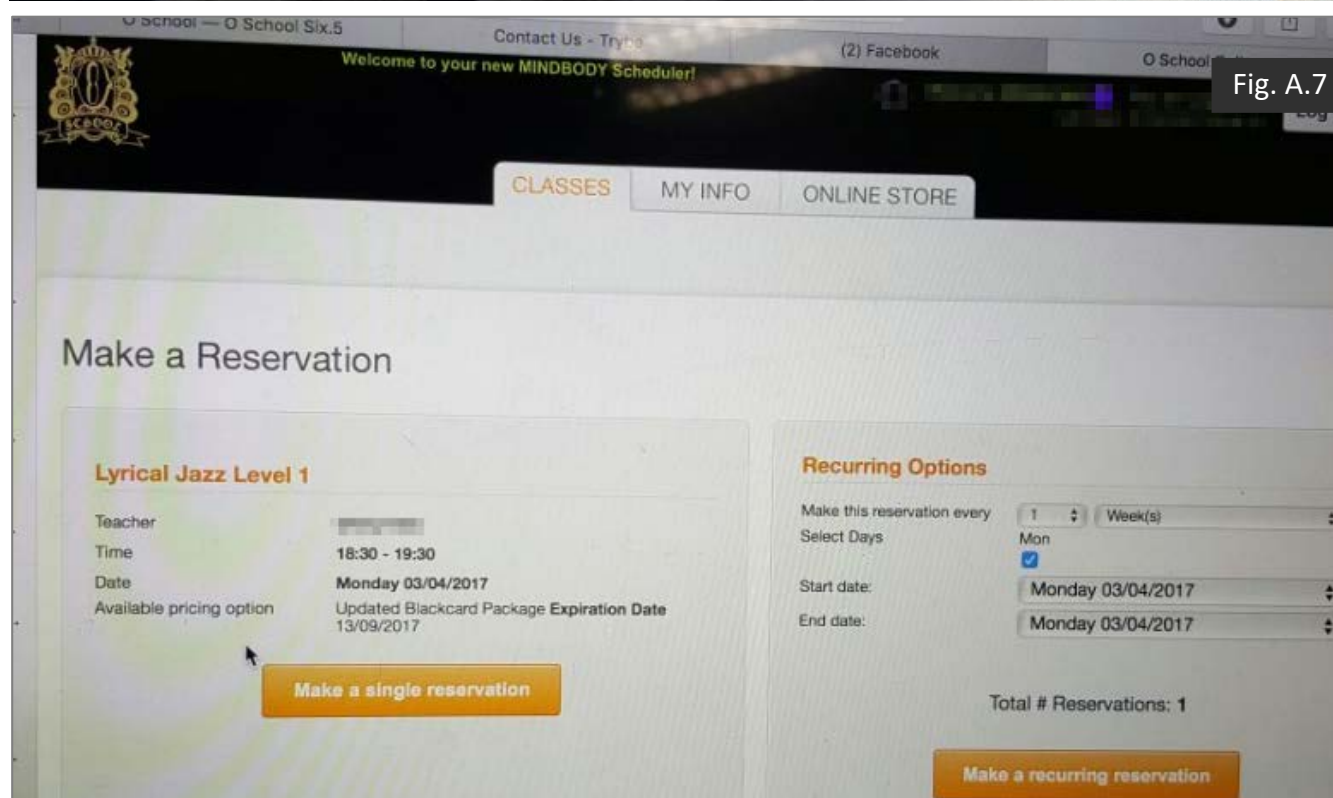
SELECT A PAYMENT METHOD:

Amount Remaining: 14 \$ Please select a payment method

Send Email Receipt? ☒ @lyq@rocketmail.com

03/04/2017 12:03 in SINGAPORE Privacy Policy Terms of Service ©2017 MINDBODY Inc. O School - Site ID: 196190 \*Scope 2 Orchard Link, Singapore 237678 Powered by O MINDBODY.

Fig. A.7 .1



**CLASSES** MY INFO ONLINE STORE

Welcome to your new MINDBODY Scheduler!

**Make a Reservation**

**Lyrical Jazz Level 1**

Teacher: [Redacted]  
Time: 18:30 - 19:30  
Date: Monday 03/04/2017  
Available pricing option: Updated Blackcard Package Expiration Date 13/09/2017

**Recurring Options**

Make this reservation every: 1 Week(s)  
Select Days: Mon ☒  
Start date: Monday 03/04/2017  
End date: Monday 03/04/2017

Total # Reservations: 1

Fig. A.7 2

**Fig. A.7 3 and Fig. A.7 4 Take as many photos as possible to gather information. The outcome is mapping out the business domain processes, business domain knowledge about the customer.**

**Without business domain knowledge,** it is impossible to help customers make good decisions to improve their business processes. To summarize, enough information to *think like them* which leads to *you can feel their pain and their needs*.

Fig. A.7 3





## What is business domain knowledge?

The term **business domain knowledge** is important to you when you position yourself as a solution developer in final year project. This reference at <https://www.boxuk.com/insight/blog-posts/importance-domain-knowledge> has a good summary.

# The importance of domain knowledge

## The importance of domain knowledge

Domain knowledge is incredibly valuable for software development. How can you ensure your external teams have this vital skill?

### The importance of domain knowledge

If you're part of an organisation where software is key to the successful running of your business, there are probably lots of areas where you'll engage people to work with it, from writing code and designing interfaces to research, maintenance and support. Of all the things these people can offer your business, one of the most valuable is **domain knowledge**; an in-depth understanding of your business, process and industry.

Gathering this knowledge can be an expensive process, requiring an investment of time and money, as well as potentially incurring efficiency and productivity losses as you bring people up to speed. How then can you ensure that your development teams have the level of domain knowledge you need while ensuring your budget doesn't [spiral out of control](#)?

End