**Object-Oriented Analysis and Design**

**Grab Your Time**

**Homework #6**

|  |  |  |
| --- | --- | --- |
| Name | ID | E-mail |
| 鍾承翰 | 106598024 | cmf15bsw@gmail.com |
| 楊子冊 | 106598054 | tzutse.yang@gmail.com |
| 吳彥銘 | 106598060 | 40143109wu@gmail.com |

目錄

[目錄 2](#_Toc515612972)

[**1.** **Requirement Document** 3](#_Toc515612973)

[1.1 Change History 3](#_Toc515612974)

[1.2 Problem Statement 3](#_Toc515612975)

[1.3 System Context Diagram 5](#_Toc515612976)

[1.4 New System Features 5](#_Toc515612977)

[1.5 Use Case Diagram 5](#_Toc515612978)

[1.6 Use Cases 6](#_Toc515612979)

[1.7 Non-functional Requirements and Constraints 11](#_Toc515612980)

[1.8 Glossary 11](#_Toc515612981)

[1.9 Software Environments 11](#_Toc515612982)

[**2.** **Domain Model** 11](#_Toc515612983)

[2.1 Concept 12](#_Toc515612984)

[2.2 Add Associations 13](#_Toc515612985)

[2.3 Add Attributes 14](#_Toc515612986)

[2.4 Combine with the original domain model derived in iteration I. 15](#_Toc515612987)

[**3.** **Design** 15](#_Toc515612988)

[3.1 Logical Architecture 16](#_Toc515612989)

[3.2 Use-Case Realizations with GRASP Patterns 17](#_Toc515612990)

[3.2.1 System Sequence Diagram 17](#_Toc515612991)

[3.2.2 Operation Contract 19](#_Toc515612992)

[**4.** **Implementation Class Model** 24](#_Toc515612993)

[**5.** **Programming** 26](#_Toc515612994)

[**6.** **Unit Testing** 36](#_Toc515612995)

[**7.** **Measurement** 41](#_Toc515612996)

1. **Requirement Document**
   1. Change History

|  |  |  |
| --- | --- | --- |
| Revision | Description | Date |
| 1 | Problem statement | 03/09 |
| 2 | Problem Statement refinement.  System Context Diagram.  Summary of System Features.  Use Case Diagram.  Use Cases.  Non-functional Requirements and Constraints.  Glossary.  Software Environments refinement. | 03/23 |
| 3 | Domain Class Diagram.  Add Associations.  Add Attributes. | 04/03 |
| 4 | Logic Architecture Diagram.  System Sequence Diagram.  System Operation.  Class Diagram.  Modify Domain Model. | 4/16、4/18、4/20 |
| 5 | Remove the Use Case – import.  semester timetable.  Modify whole the document . | 4/24 |
| 6 | Add System Sequence Diagram Add Sequence Diagram | 5/9 |
| 7 | Modify Context Diagram  Modify Use Case Diagram  Modify Domain Model  Modify Class Diagram | 5/10 |
| 8 | Add Source Code of Project  Add Source Code of Test Case of Project | 5/11 |
| 9 | Modify Use Case  Modify Domain Model  Add “Combine with the original domain model derived in iteration I.” | 5/31 |

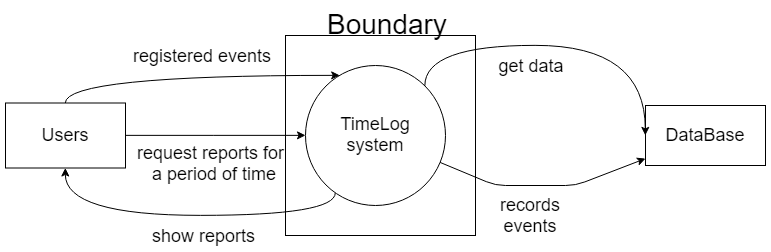
* 1. Problem Statement

在求學過程中，我們往往不知道該如何有效運用時間，而最根本的原因就是不確定將時間花在哪裡，而渾渾噩噩地過著日復一日的生活。因此我們需要有一個軟體能夠幫我們紀錄除了上課時間之外還做了哪些事情，並最後可以用簡單的圖表來呈現讓使用者可以知道時間都花在甚麼事情上，進一步去調整每件事情時間的分配。

對於北科學生們來說，可以藉由此軟體記錄下自己做的事情及時間。如果你想知道每段時間做了什麼，花了多少時間，只要設定日期範圍，即可產生相對應的圖表。

圖表顯示出這段時間內每項事情及其所花費的時數，藉此思考事情的規劃到底是不是有效率的。

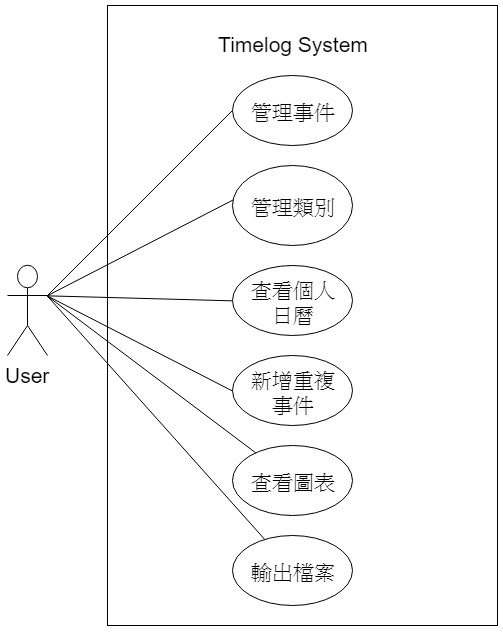
* 1. System Context Diagram



* 1. New System Features

|  |  |
| --- | --- |
| **Feature ID** | **Description** |
|  | 管理事件 |
|  | 搜尋事件 |
|  | 查看圖表 |
|  | 輸出檔案 |
|  | 匯入課表 |
|  | 管理類別 |

* 1. Use Case Diagram



* 1. Use Cases

|  |  |
| --- | --- |
| Use Case ID |  |
| Use Case Name | 管理事件 |
| Scope | Timelog System. |
| Level | User goal. |
| Primary Actor | User. |
| Stakeholders and Interests | User：使用者想管理事件。 |
| Preconditions | 1. 登入系統. 2. 修改與刪除至少一筆事件。 |
| Success Guarantee | 系統成功紀錄在事件清單中。 |
| Main Success Scenario | 1. 使用者登入系統。 2. 使用者進入管理事件頁面，並顯示事件清單。 3. 使用者選擇管理事件。 4. 管理事件：    1. 使用者新增事件，隨即輸入事件描述、設定所屬類別和紀錄事件花費的時間。    2. 使用者選擇欲編輯的事件，隨即更改事件描述、所屬類別或事件花費的時間。    3. 使用者選擇欲刪除的事件並點選刪除。 5. 完成管理事件：    1. 使用者選擇儲存事件。    2. 使用者選擇編輯完成。 6. 系統更新事件清單並儲存。 7. 系統會在畫面上顯示最新的事件清單。 |
| Extensions | 1. 無法連線至網路： 2. 使用者連線至網路。 3. 沒有選擇事件類別： 4. 使用者選擇類別。 5. 起始時間大於結束時間： 6. 使用者設定正確起始與結束時間。 7. 沒有選擇事件類別： 8. 使用者選擇類別。 9. 起始時間大於結束時間： 10. 使用者設定正確起始與結束時間。 |
| Special Requirements | None. |
| Technology and Data Variations List | None. |
| Frequency of Occurrence | Always. |
| Open Issues | None. |

|  |  |
| --- | --- |
| Use Case ID |  |
| Use Case Name | 搜尋事件 |
| Scope | Timelog System. |
| Level | User goal. |
| Primary Actor | Student. |
| Stakeholders and Interests | Student：使用者設定搜尋條件來尋找事件。 |
| Preconditions | 系統存在一個以上的事件。 |
| Success Guarantee | 系統會在頁面上顯示搜尋結果。 |
| Main Success Scenario | 1. 使用者登入系統。 2. 使用者進到搜尋頁面。 3. 使用者設定搜尋條件。 4. 系統根據條件來搜尋事件。 5. 系統在頁面顯示搜尋結果。 |
| Extensions | 1. 無法連線至網路： 2. 系統持續偵測網路。 3. 使用者輸入不存在的搜尋項目： 4. 系統頁面會顯示空白的結果清單。 5. 使用者輸入的條件不符合格式： 6. 系統會通知使用者輸入不符合格式時通知使用者並提示使用者正確的格式。 7. 使用者輸入空白或者未輸入任何東西： 8. 系統頁面會顯示空白的搜尋結果。 9. 使用者取消行為： 10. 系統會暫存使用者設定的搜尋條件。 11. 使用者切換至其他頁面。 |
| Special Requirements | None. |
| Technology and Data Variations List | None. |
| Frequency of Occurrence | Sometimes. |
| Open Issues | None. |

|  |  |
| --- | --- |
| Use Case ID |  |
| Use Case Name | 查看圖表 |
| Scope | Timelog System. |
| Level | User goal. |
| Primary Actor | Student. |
| Stakeholders and Interests | Student：使用者想要觀看事件的統計圖。 |
| Preconditions | 系統存在一個以上的事件。 |
| Success Guarantee | 系統將統計結果以統計圖的方式顯示在畫面上。 |
| Main Success Scenario | 1. 使用者登入系統。 2. 使用者進到統計頁面。 3. 使用者選擇日期範圍。 4. 系統計算每項類別在日期範圍內所花費的總時數。 5. 系統將統計結果以圖表方式呈現在預覽畫面上。 |
| Extensions | 1. 無法連線至網路： 2. 系統持續偵測網路。 3. 使用者選擇不符合實際的日期範圍： 4. 系統在偵測到時間範圍的起始時間小於終止時間時會傳訊息要求使用者輸入符合現實條件的時間範圍。 5. 使用者沒有類別在類別清單： 6. 沒有類別進行計算，將整體總計時間設為0。 7. 使用者沒有類別在類別清單： 8. 沒有類別進行計算，統計圖表顯示的時間刻度照常，但類別的顯示為空，顯示總計時間為0。 9. 使用者沒有事件在事件清單，但有類別在類別清單： 10. 沒有事件進行計算，統計圖表顯示的時間刻度照常，但類別的顯示類別清單項目，顯示各類別總計時間為0。 |
| Special Requirements | None. |
| Technology and Data Variations List | None. |
| Frequency of Occurrence | Always. |
| Open Issues | None. |

|  |  |
| --- | --- |
| Use Case ID |  |
| Use Case Name | 輸出檔案 |
| Scope | Timelog System. |
| Level | User goal. |
| Primary Actor | Student. |
| Stakeholders and Interests | Student：使用者想要將統計結果以檔案的方式儲存在自己的裝置上。 |
| Preconditions | 使用者已產生統計圖表。 |
| Success Guarantee | 使用者成功的將檔案下載到裝置中。 |
| Main Success Scenario | 1. 使用者已至統計頁面並產生統計圖表。 2. 使用者選擇輸出的檔案類型。 3. 系統根據使用者所選擇的檔案類型，將統計圖表輸出，並顯示下載連結。 4. 使用者進行下載。 |
| Extensions | 1. 無法連線至網路： 2. 系統持續偵測網路。 3. 使用者沒有產生統計圖表： 4. 系統會禁止使用者選擇輸出檔案類型。 |
| Special Requirements | None. |
| Technology and Data Variations List | None. |
| Frequency of Occurrence | Always. |
| Open Issues | None. |

|  |  |
| --- | --- |
| Use Case ID |  |
| Use Case Name | 管理類別。 |
| Scope | Timelog System. |
| Level | User goal. |
| Primary Actor | User. |
| Stakeholders and Interests | User：使用者想管理類別。 |
| Preconditions | None. |
| Success Guarantee | 系統成功更新類別清單。 |
| Main Success Scenario | 1. 使用者登入系統。 2. 使用者進入管理事件頁面，並顯示事件清單。 3. 使用者選擇管理事件。 4. 管理類別：    1. 使用者新增類別，隨即輸入類別描述。    2. 使用者選擇欲編輯的類別，隨即更改類別描述。    3. 使用者選擇欲刪除的類別並刪除。 5. 完成管理事件：    1. 使用者選擇儲存事件。    2. 使用者選擇編輯完成。 6. 系統更新事件清單並儲存。 7. 系統顯示最新的類別清單。 |
| Extensions | 1. 無法連線至網路： 2. 使用者連線至網路。 3. 沒有設定類別資訊： 4. 設定類別資訊。 5. 重複類別名稱： 6. 重新設定類別名稱。 7. 沒有選擇類別： 8. 使用者選擇類別。 9. 編輯後的名稱與既存在的類別名稱相同 10. 重新編輯類別名稱。 |
| Special Requirements | None. |
| Technology and Data Variations List | None. |
| Frequency of Occurrence | Sometimes. |
| Open Issues | None. |

|  |  |
| --- | --- |
| Use Case ID |  |
| Use Case Name | 新增重複事件 |
| Scope | Timelog System. |
| Level | User goal. |
| Primary Actor | User. |
| Stakeholders and Interests | User：使用者想要紀錄固定時間會發生的事件在事件清單中。 |
| Preconditions | None. |
| Success Guarantee | 系統成功更新事件清單。 |
| Main Success Scenario | 1. 使用者登入系統。 2. 使用者進入管理事件頁面，並顯示事件清單。 3. 使用者選擇管理事件。 4. 使用者新增事件，隨即輸入事件描述、設定所屬類別和紀錄事件花費的時間。 5. 使用者將事件設定為重複事件，設定事件週期，並設定起始時間與結束時間。 6. 使用者完成新增重複事件。 7. 系統更新事件清單並儲存。 |
| Extensions | 1. 無法連線至網路： 2. 使用者連線至網路。 3. 沒有選擇事件類別： 4. 使用者選擇類別。 5. 起始時間大於結束時間： 6. 使用者設定正確起始與結束時間。 7. 未將事件設定為重複事件： 8. 使用者將事件設定為重複事件。 9. 未設定事件週期： 10. 系統提醒使用者需設定事件週期。 11. 使用者設定事件週期。 12. 起始日期大於結束日期： 13. 使用者設定正確起始與結束日期。 |
| Special Requirements | None. |
| Technology and Data Variations List | None. |
| Frequency of Occurrence | Sometimes. |
| Open Issues | None. |

* 1. Non-functional Requirements and Constraints

|  |  |  |
| --- | --- | --- |
| **NFR ID** | **Category** | **Description** |
|  | Response Time | 管理事件時，要立刻反應出結果。 |
|  | Performance | 產生圖表時，應小於五秒。 |
|  | Usability | 此系統對於使用者必須容易學習及使用。 |
|  | Reliability | 檔案讀寫過程中，須確保資料轉換的正確性。 |
|  | Performance | 存取檔案必須在10秒內完成。 |
|  | Performance | 系統必須能夠允許多個使用者同時登入。 |

* 1. Glossary

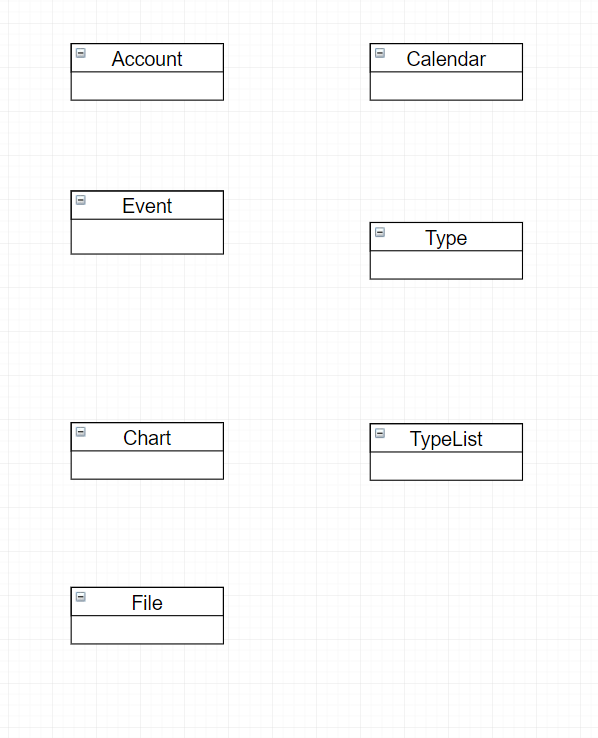
|  |  |
| --- | --- |
| **Item** | **Definition or Description** |
| 事件<Event> | 使用者紀錄過去某段時間所作的事情，並成功紀錄在事件清單中。 |
| 類別<Type> | 使用者可自訂類別名稱，以便分類事件。 |
| 圖表<Chart> | 系統會根據使用者所選擇的日期範圍來產生統計圖，而圖表內容為使用者在各類別花費的總時數。 |
| 檔案<File> | 紀錄事件的檔案，具有特定的格式，可以輸出成Png。 |

* 1. Software Environments

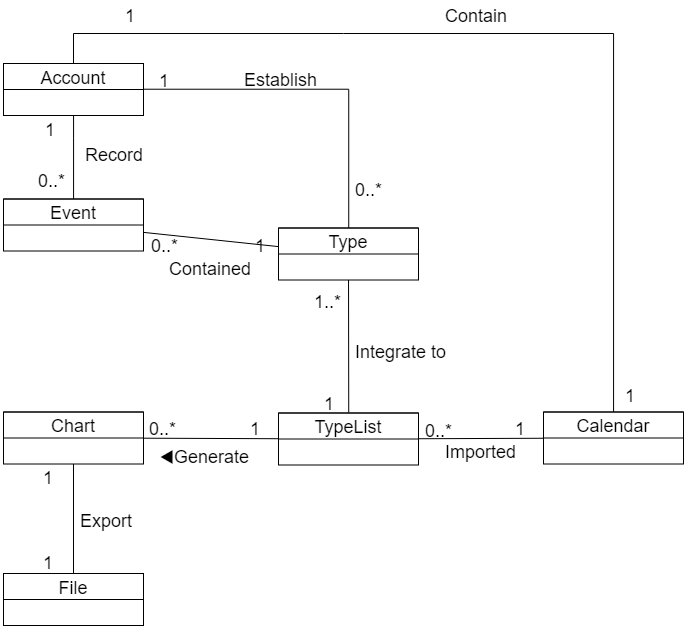
This project will be written in Javascript (ES6) language.

1. **Domain Model**
   1. Concept

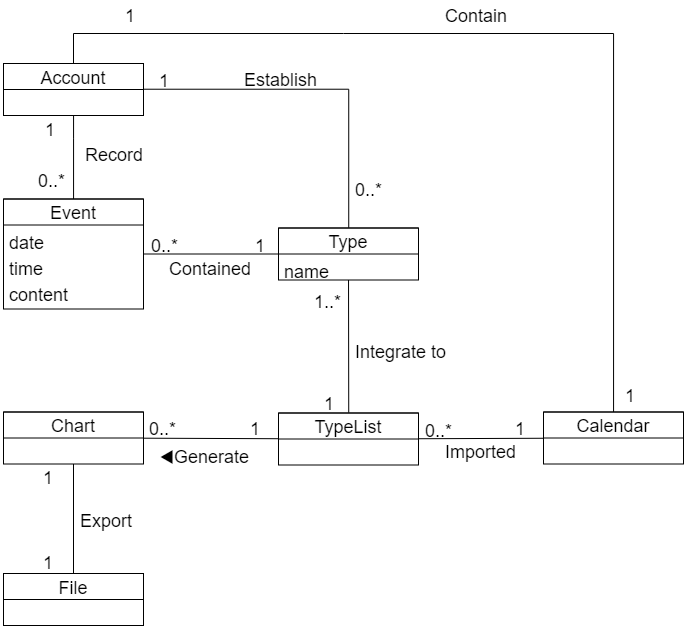
Grab Your Time是一個可以讓使用者(User)紀錄時間的系統。紀錄事件(Event)之前，使用者可以自訂類別(Type)，來分類事件類型；而要紀錄事件的時候，需要選擇類別、填寫事件內容及事件發生的時間，紀錄成功之後就會添增至事件清單(EventList)中，之後便可以設定日期範圍來計算你這段時間內各個事件所花費的總時數，並以統計圖表(Chart)來顯示。看完統計圖表後就可以選擇檔案類型(File)輸出至裝置內。



* 1. Add Associations



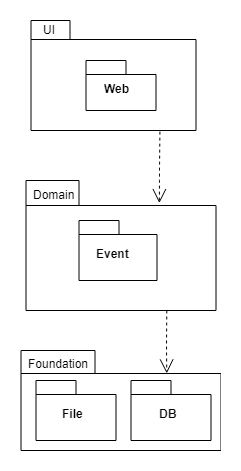
* 1. Add Attributes



* 1. Combine with the original domain model derived in iteration I.

|  |  |
| --- | --- |
| Before |  |
| After |  |

1. **Design**
   1. Logical Architecture

****

* 1. Use-Case Realizations with GRASP Patterns
     1. System Sequence Diagram

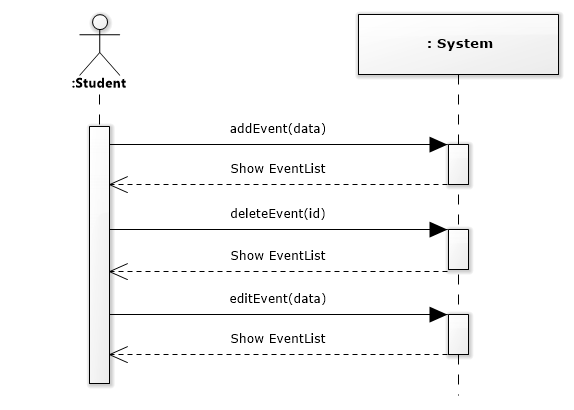


Figure 3-1 System Sequence Diagram of use case: Manage Event.

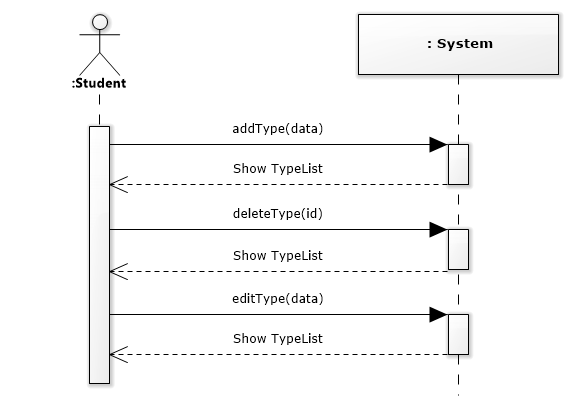
****

Figure 3-2 System Sequence Diagram of use case: Manage Type.

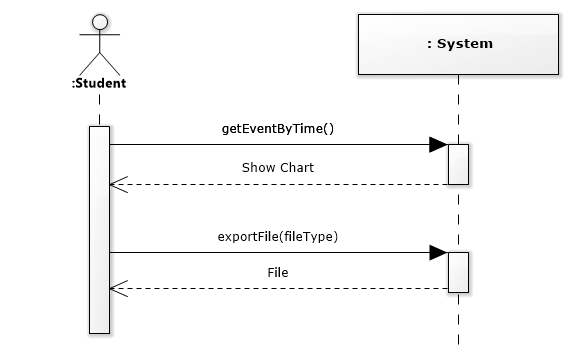


Figure 3-3 System Sequence Diagram of use case: Export File.

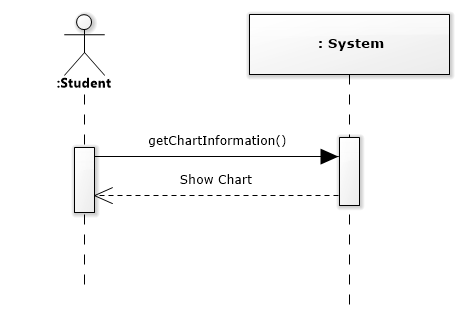


Figure 3-4 System Sequence Diagram of use case: Present Chart.

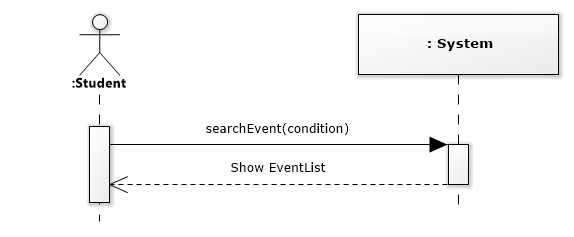


Figure 3-5 System Sequence Diagram of use case: Search Event.

* + 1. Operation Contract

|  |  |
| --- | --- |
| Contract ID | Operation Name |
|  | addEvent() |
|  | deleteEvent() |
|  | editEvent() |
|  | addType() |
|  | deleteType() |
|  | editType() |
|  | getEventByTime() |
|  | exportFile() |
|  | importTimeTable() |
|  | searchEvent() |

* + - 1. addEvent

|  |  |
| --- | --- |
| Operation | addEvent(data) |
| Cross References | Manage Event. |
| Preconditions | Login to system.  At least one type. |
| Postconditions | An event was created.  Show calendar with added event. |

* + - 1. deleteEvent

|  |  |
| --- | --- |
| Operation | deleteEvent(id) |
| Cross References | Manage Event. |
| Preconditions | Login to system. |
| Postconditions | An event was deleted.  Show calendar without the event which is deleted. |

* + - 1. editEvent

|  |  |
| --- | --- |
| Operation | editEvent (data) |
| Cross References | Manage Event. |
| Preconditions | Login to system. |
| Postconditions | An event was edit.  Show calendar with edited event. |

* + - 1. addType

|  |  |
| --- | --- |
| Operation | addType (data) |
| Cross References | Manage Type. |
| Preconditions | Login to system. |
| Postconditions | An type was created.  Show typelist with added type. |

* + - 1. deleteType

|  |  |
| --- | --- |
| Operation | deleteType (id) |
| Cross References | Manage Type. |
| Preconditions | Login to system. |
| Postconditions | An type was deleted.  Show typelist without the type which is deleted. |

* + - 1. editType

|  |  |
| --- | --- |
| Operation | editType(data) |
| Cross References | Manage Type. |
| Preconditions | Login to system. |
| Postconditions | An type was edit.  Show typelist without the type which is deleted. |

* + - 1. getEventByTime

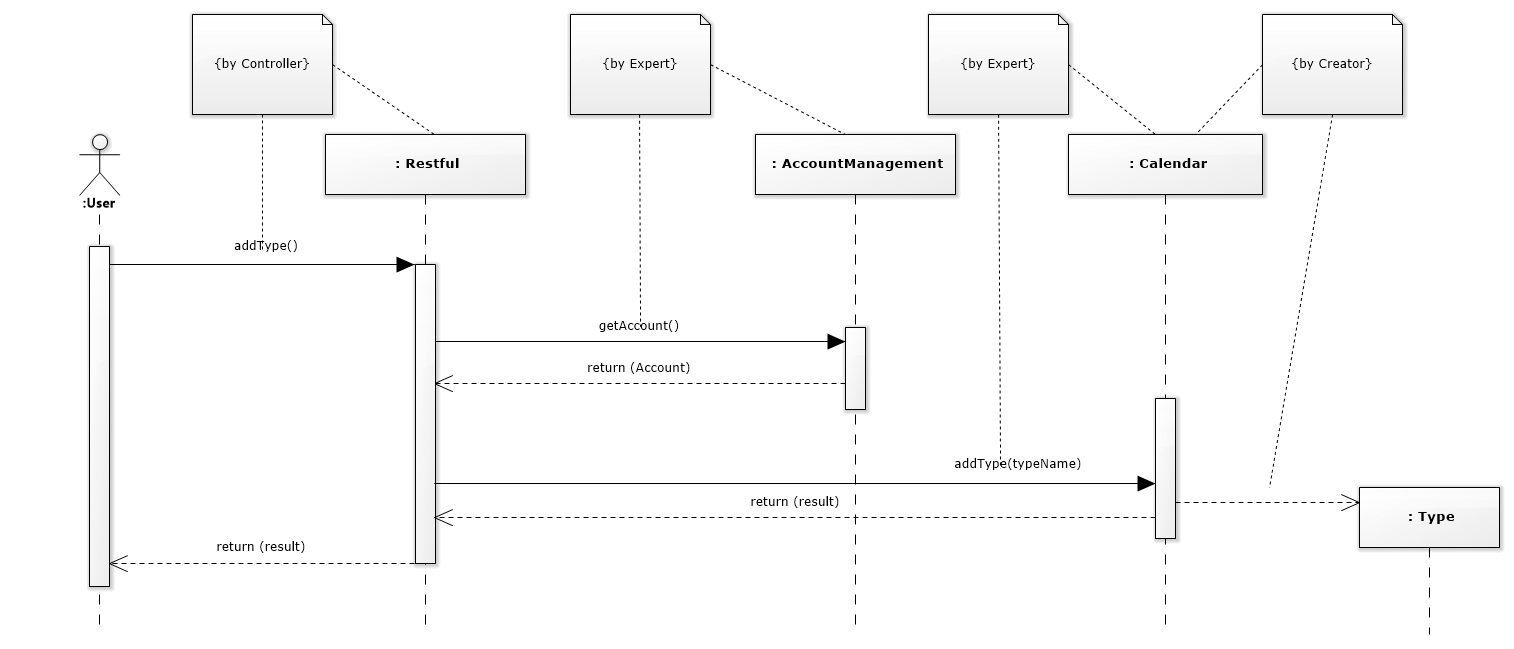
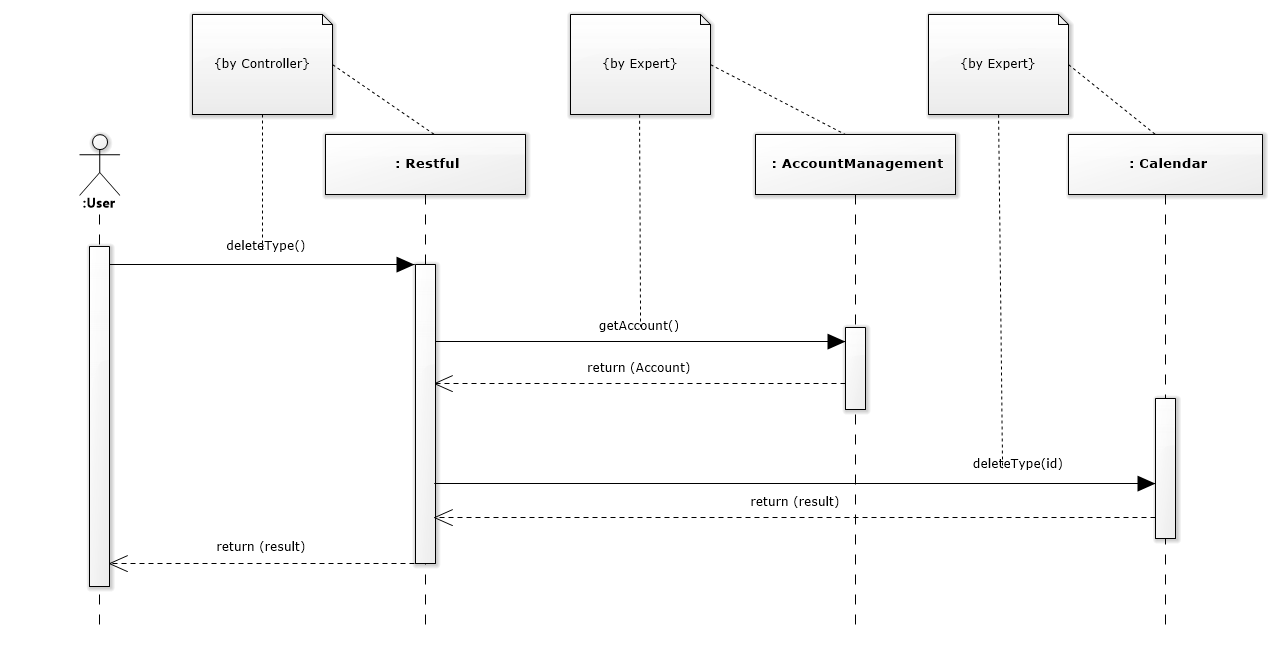
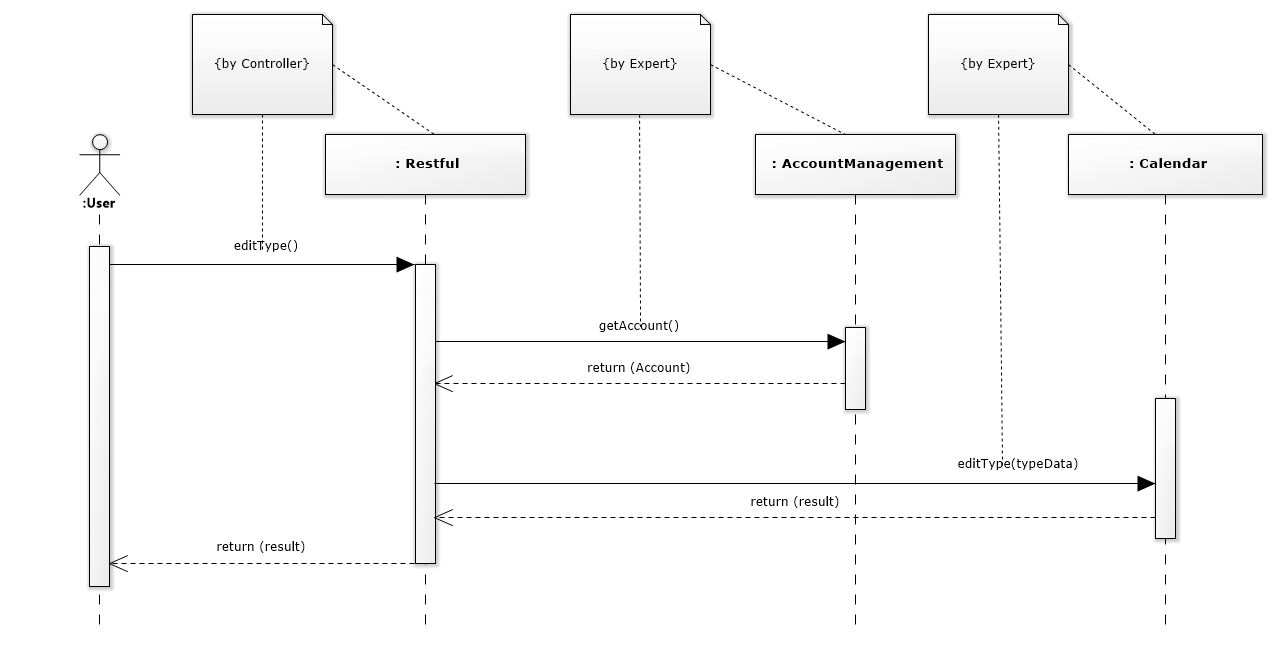
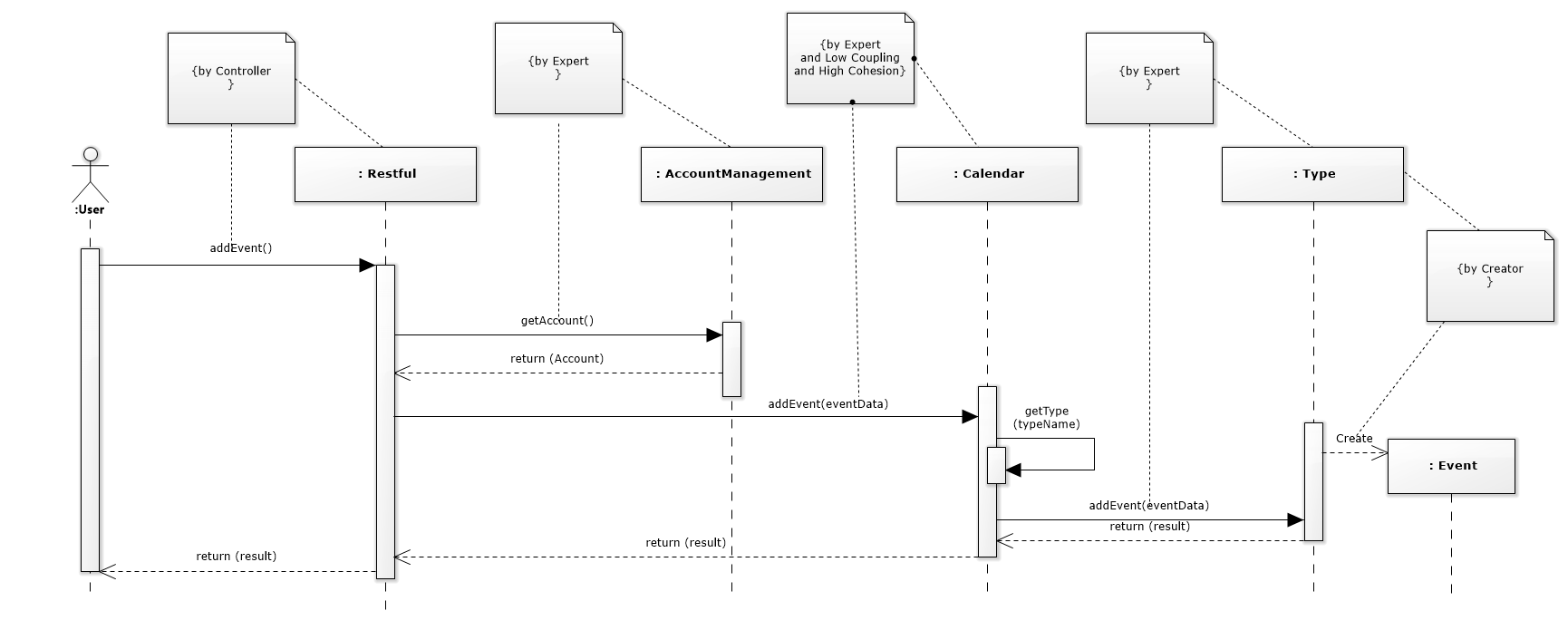
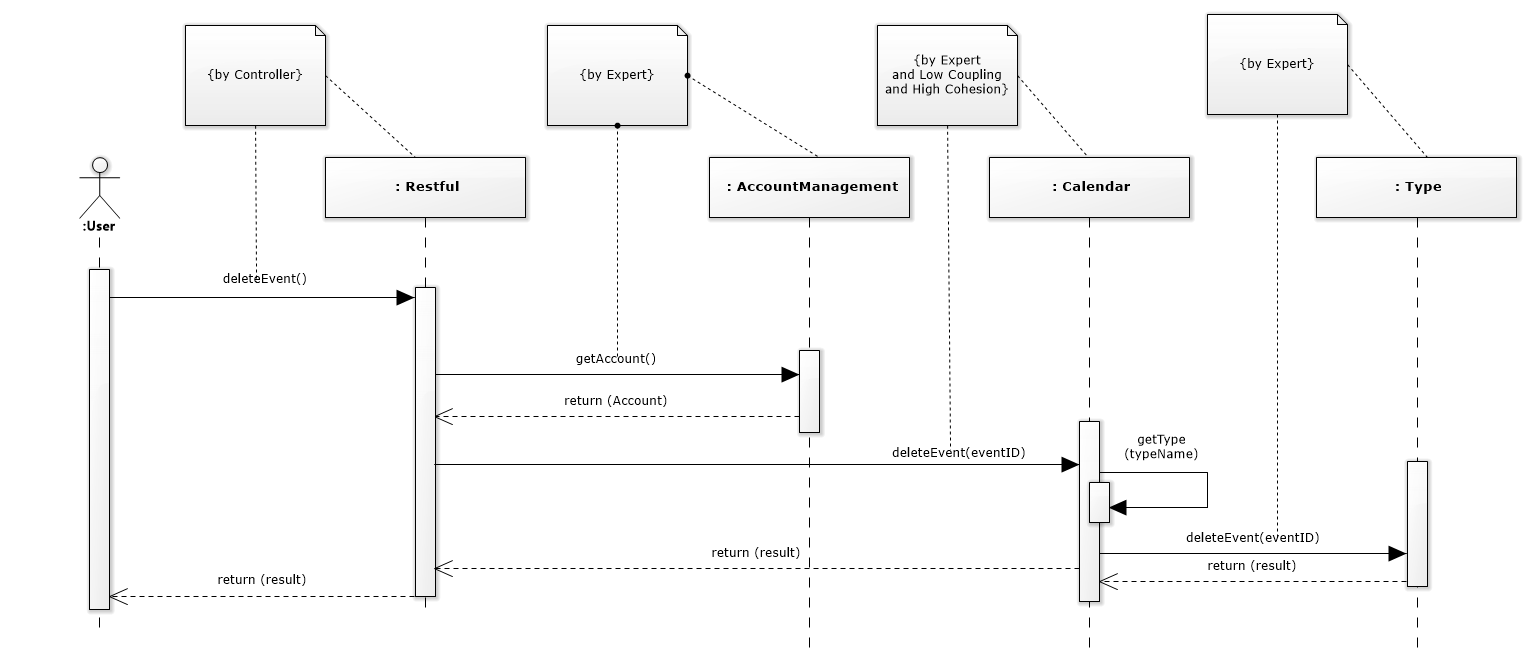
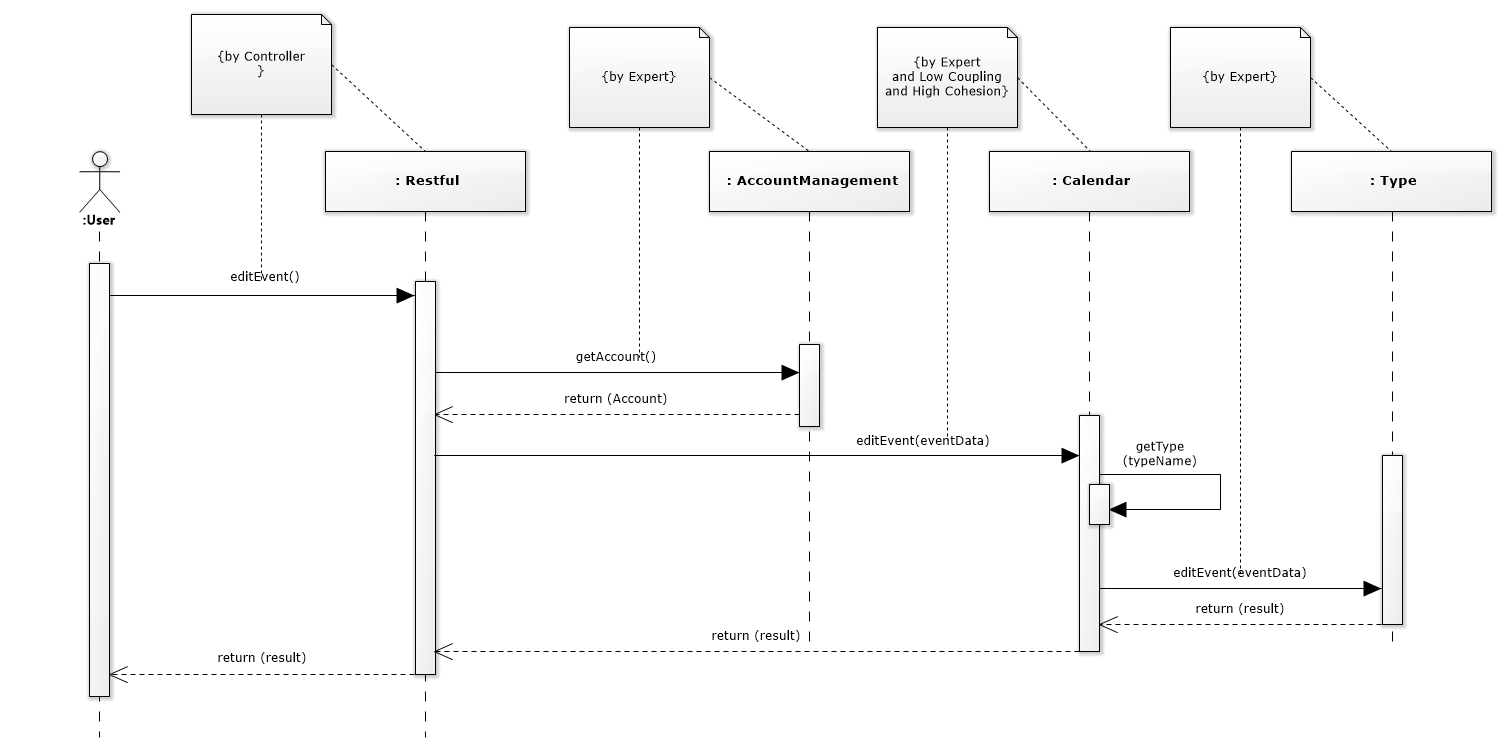
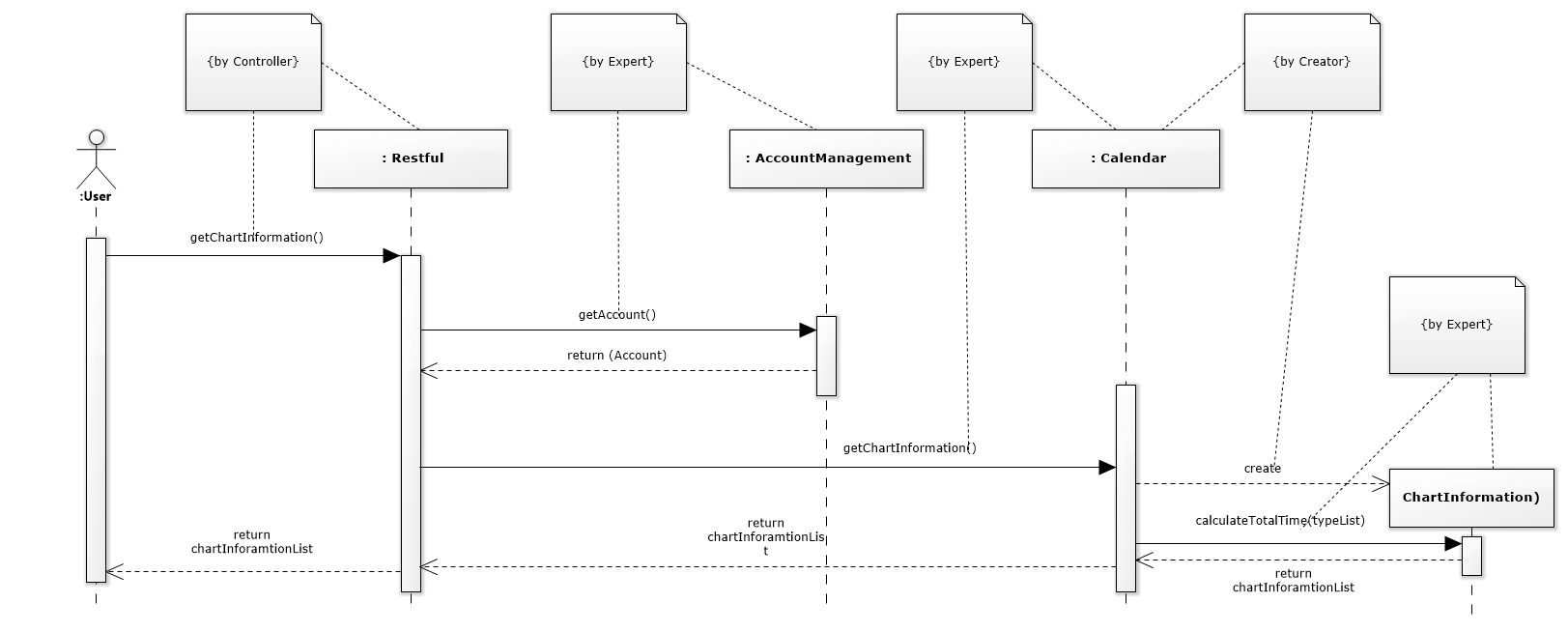
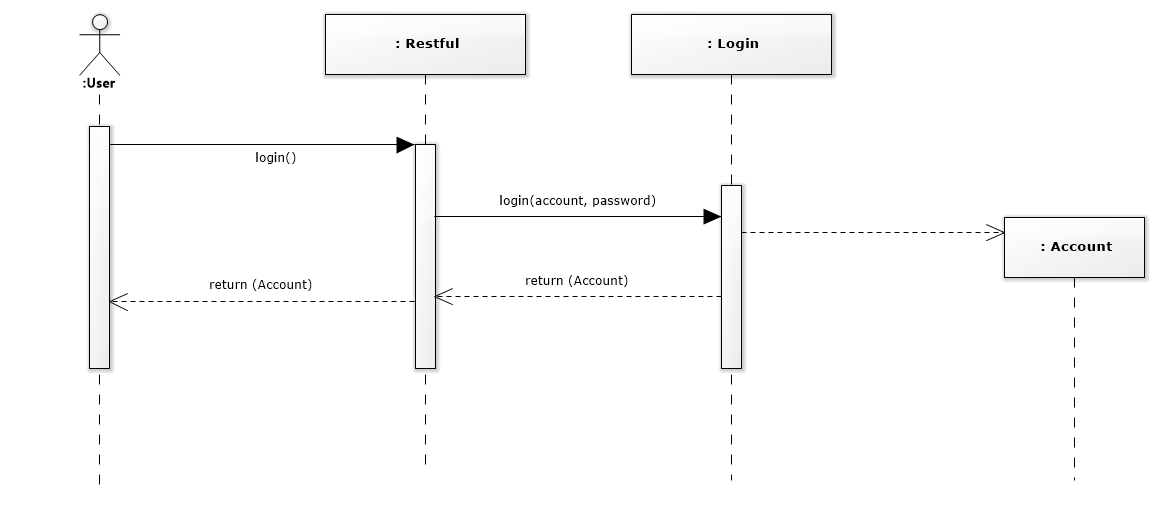
|  |  |
| --- | --- |
| Operation | getEventByTime() |
| Cross References | Present Chart. |
| Preconditions | At least one type. |
| Postconditions | Show charts with events in the time range |

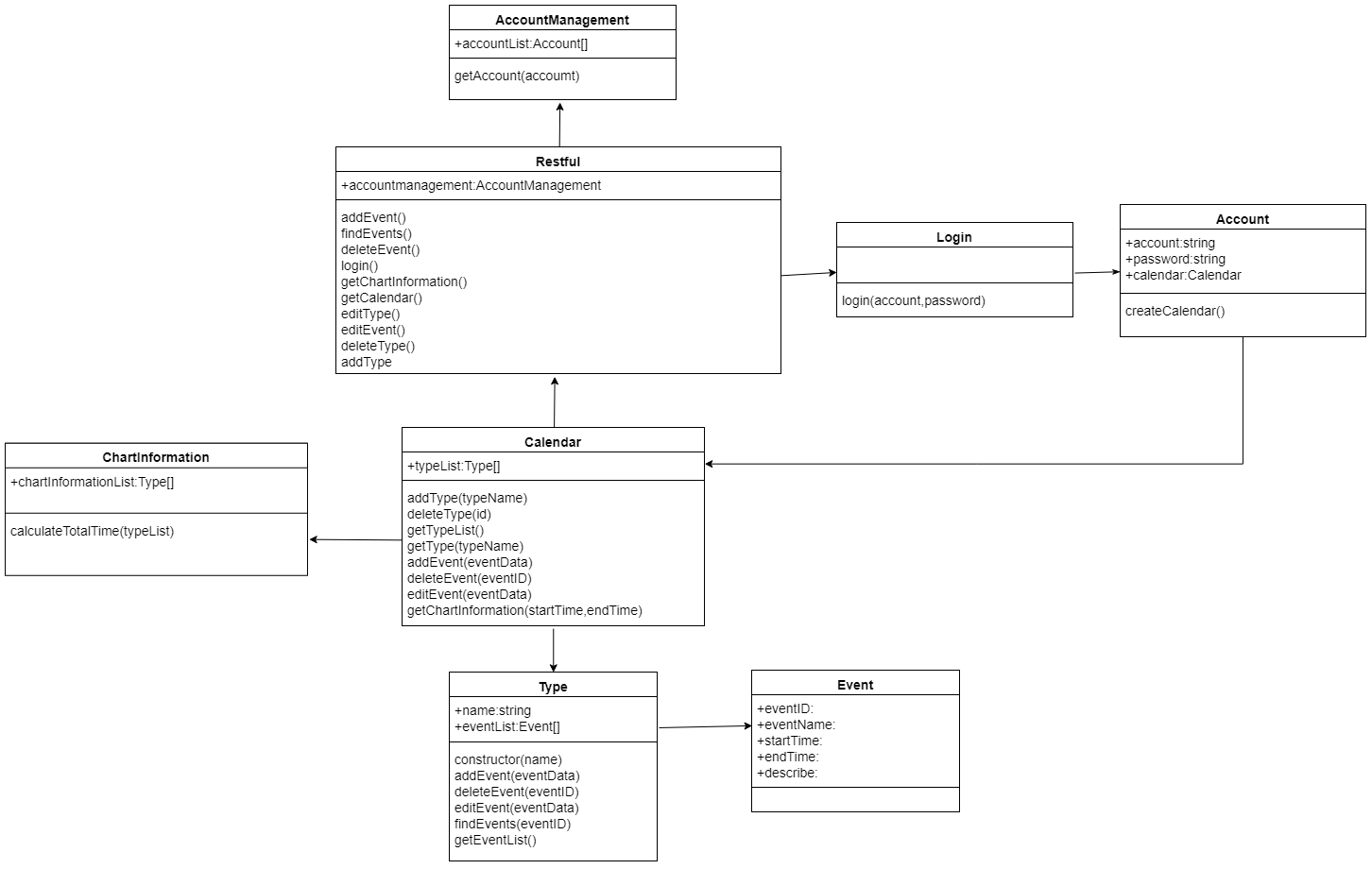
* + - 1. exportFile

|  |  |
| --- | --- |
| Operation | exportFile (fileType) |
| Cross References | Export File. |
| Preconditions | Show charts with events in the time range. |
| Postconditions | Show download page. |

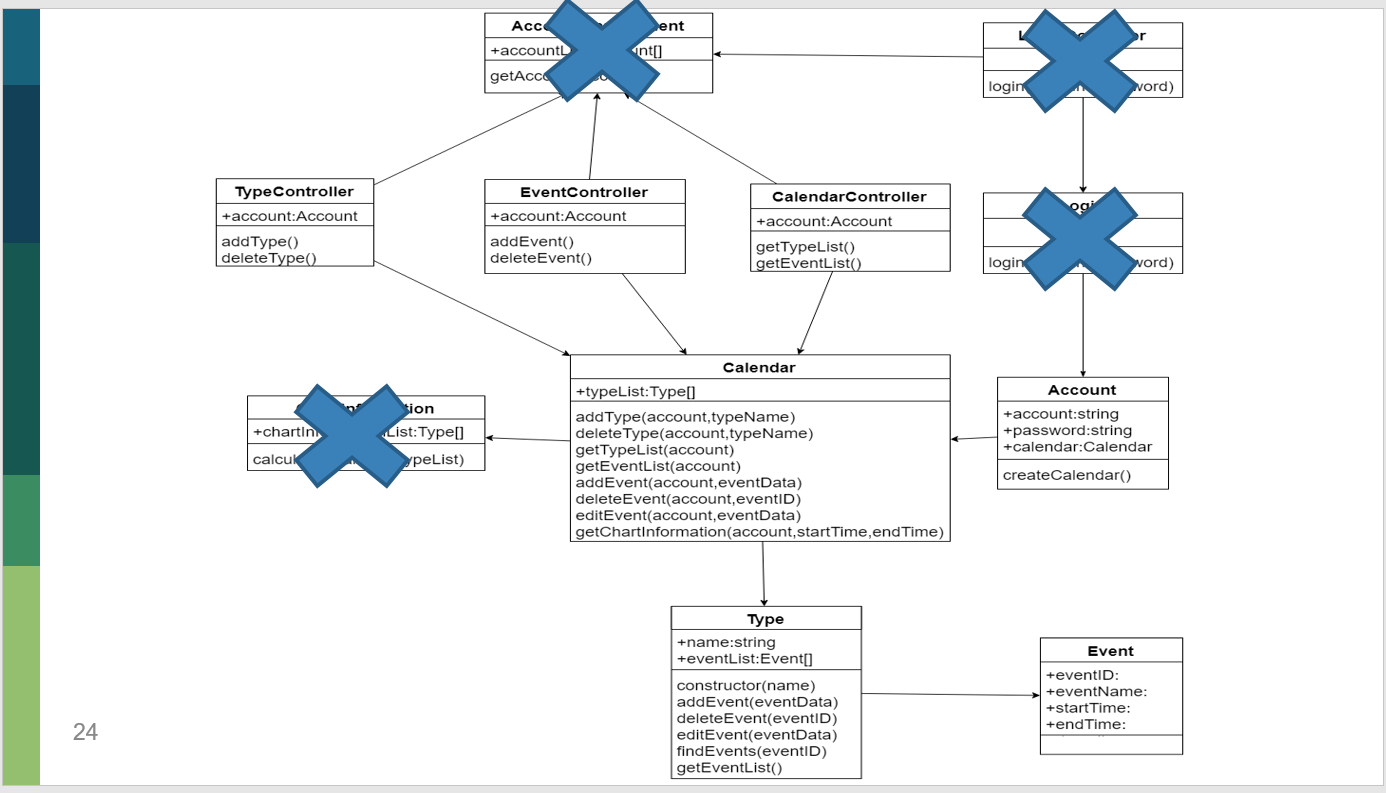
* + - 1. searchEvent

|  |  |
| --- | --- |
| Operation | searchEvent (condition) |
| Cross References | Search Event. |
| Preconditions | Login to system. |
| Postconditions | Show result eventList. |

* + 1. SD
       1. addType
       2. deleteType
       3. editType
       4. addEvent
       5. deleteEvent
       6. editEvent
       7. getChartInformation
       8. login
  1. Class Diagram



1. **Implementation Class Model** 
   1. Draw an implementation class diagram for your system



* 1. Show the difference between implementation class model and design

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Method** | **Design** | **Imp** |
| **Type** | **addEvent** | **Yes** | **Yes** |
| **deleteEvent** | **Yes** | **Yes** |
| **Calendar** | **addType** | **Yes** | **Yes** |
| **deleteType** | **Yes** | **Yes** |
| **getTypeList** | **Yes** | **Yes** |
| **addEvent** | **Yes** | **Yes** |
| **deleteEvent** | **Yes** | **Yes** |
| **getEventList** | **Yes** | **Yes** |
| **editEvent** | **Yes** | **No** |
| **editType** | **Yes** | **No** |
| **getChartInformation** | **Yes** | **No** |
| **ChartInformation** | **caculateTotalTime** | **Yes** | **No** |
| **Event** | **None** | **Yes** | **Yes** |
| **Account** | **createCalenedar** | **Yes** | **Yes** |
| **CalendarController** | **getTypeList** | **Yes** | **Yes** |
| **getEventList** | **Yes** | **Yes** |
| **EventController** | **addEvent** | **Yes** | **Yes** |
| **deleteEvent** | **Yes** | **Yes** |
| **TypeController** | **addType** | **Yes** | **Yes** |
| **deleteType** | **Yes** | **Yes** |

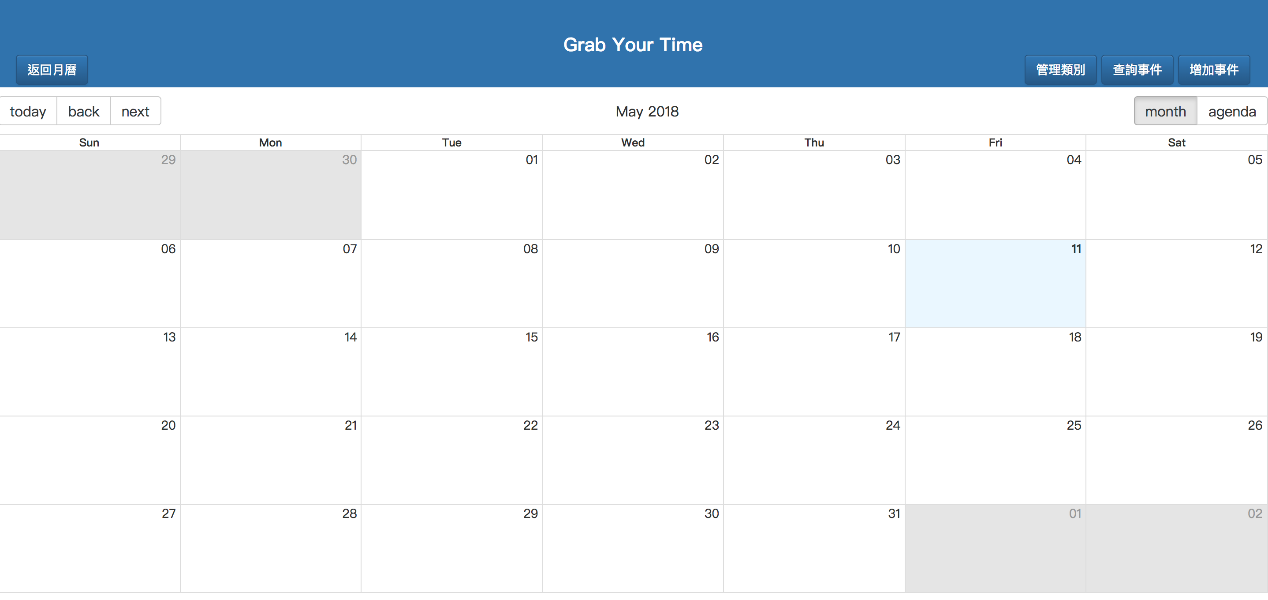
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Numner of Added** | **Numner of Removed** | **Numner of Modified** |
| **Class** | **0** | **1** | **0** |
| **Method** | **0** | **4** | **0** |

* 1. Calculate Line of Code

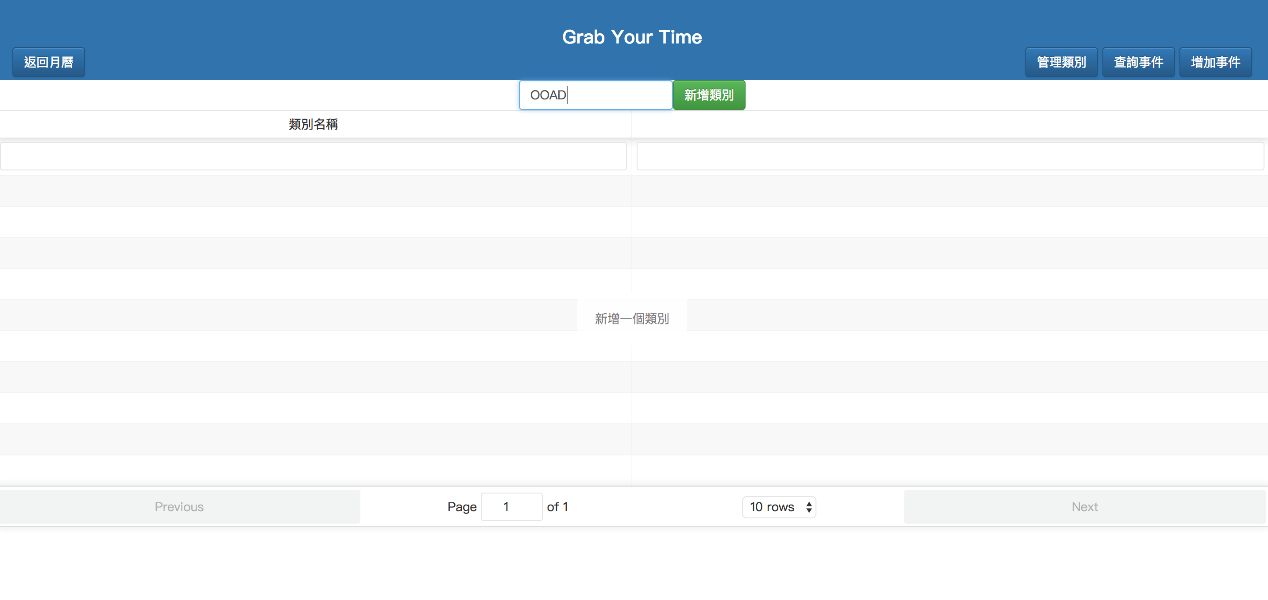
|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Numner of Added** | **Number of methods** | **Line of Code in Class** |
|  | **Type** | **2** | **46** |
|  | **Calendar** | **6** | **66** |
|  | **ChartInformation** | **1** | **0** |
|  | **Event** | **1** | **8** |
|  | **Account** | **1** | **20** |
|  | **CalendarController** | **2** | **13** |
|  | **EventController** | **2** | **19** |
|  | **TypeController** | **2** | **15** |
| **Sum** |  | **17** | **187** |

1. **Programming**
   1. Snapshots of system execution

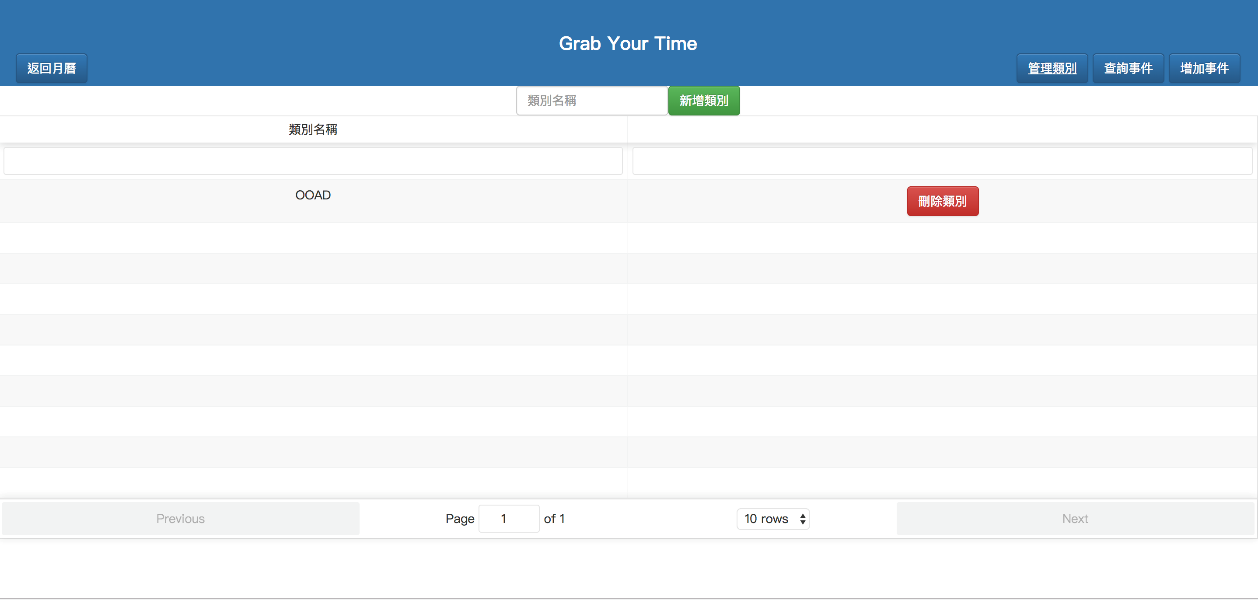
查看個人日曆

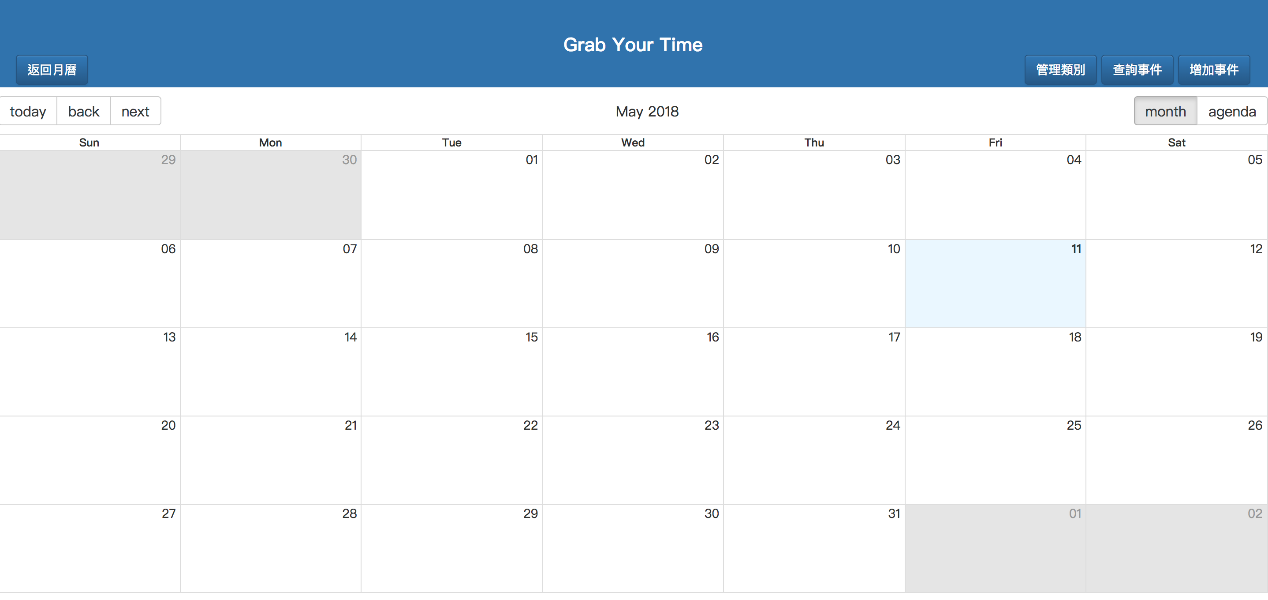


新增類別

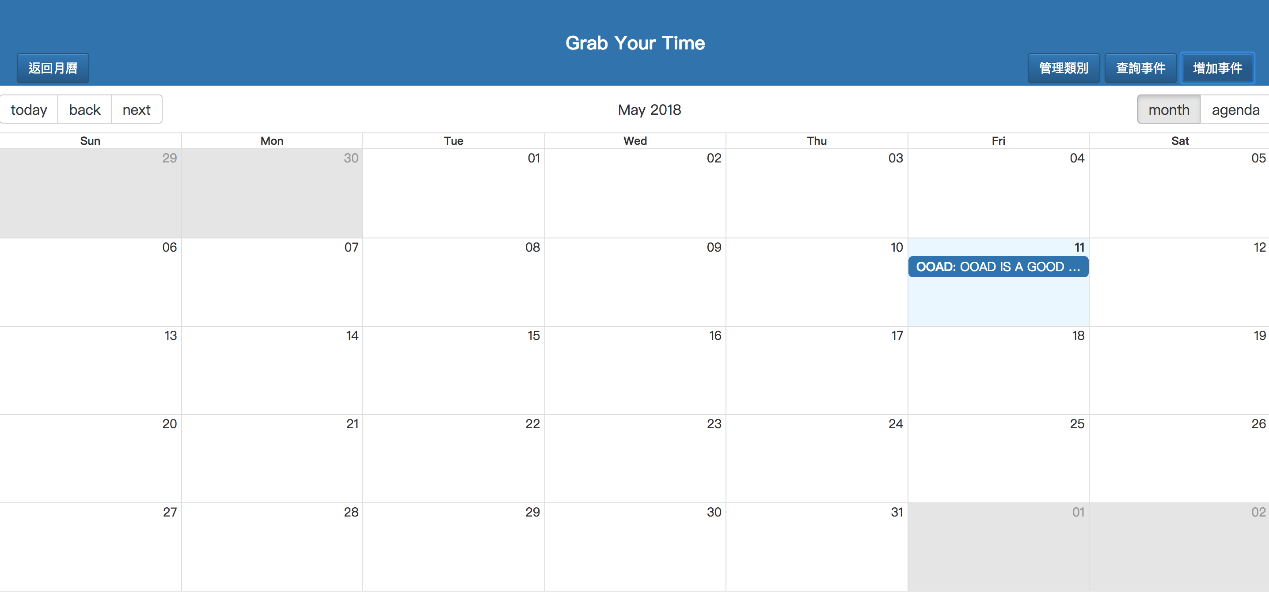


新增類別成功

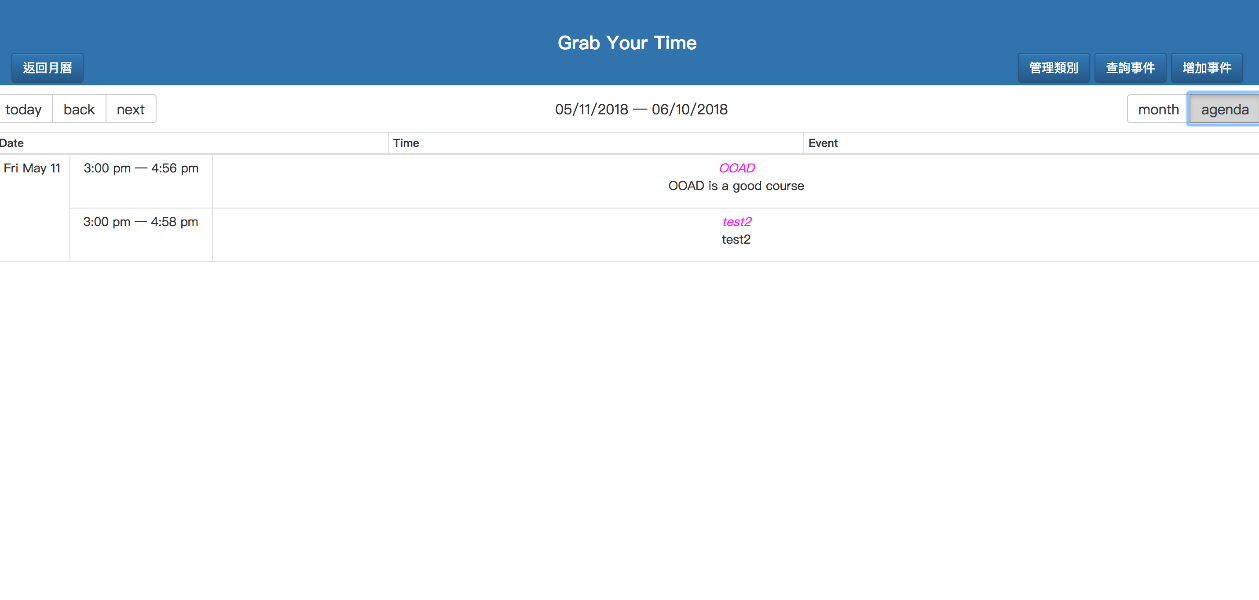


新增事件

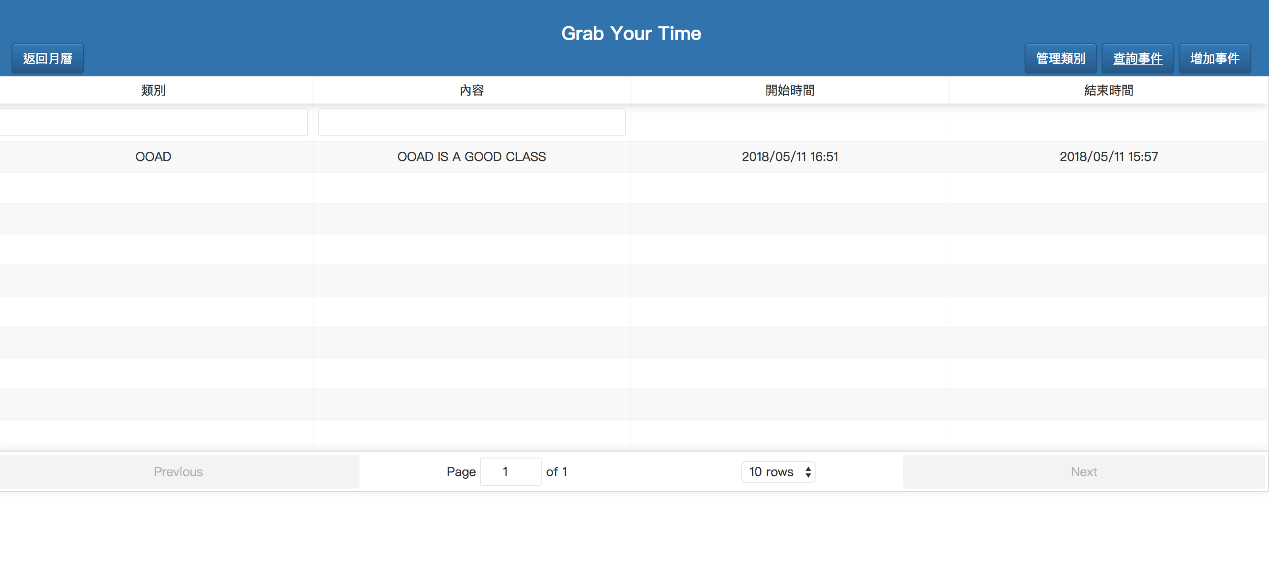
新增事件成功



新增事件成功-2



查詢事件



* 1. Source Code Listing

**Calendar.js**

|  |
| --- |
| import Account from './Account';  import ChartInformation from './ChartInformation';  import mongoose from 'mongoose';  var db = mongoose.connection;  import calendarSchema from './Schemas/calendarSchema';  import Type from './Type';  class Calendar {  constructor(typeList){  this.typeList = typeList ;  }  addType(\_account,typeObj){  if(this.typeList.map((type)=>{  return type.typeName  }).indexOf(typeObj.typeName)=== -1){  this.typeList.push(typeObj);  calendarSchema.update({account:\_account},{$set: {typeList:this.typeList}})  .then((result) => {  console.log(result);  })  }  else  console.log('same Type');  }  deleteType(\_account,typeName){  this.typeList = this.typeList.filter((type)=>{  return type.typeName !== typeName;  })  calendarSchema.update({account:\_account},{$set: {typeList:this.typeList}})  .then((result) => {  console.log(result);  })  }  getTypeList(){  return this.typeList;  }  getChartInformation(startDate,endDate){  let chartInformation = new ChartInformation()  return chartInformation.getChartInformation(this.typeList,startDate,endDate)  }  addEvent(\_account, eventData){  return new Promise((resolve, reject)=>{  let findType = this.typeList.filter((type)=>{  if(type.typeName === eventData.title)  return type  })  if(eventData.checked === true ){  findType[0].addCycleEvent(\_account, eventData, calendarSchema)  .then((result)=>{  resolve(result);  })  .catch((err)=>{  reject(err);  })  }  else{  findType[0].addEvent(\_account, eventData, calendarSchema)  .then((result)=>{  resolve(result);  })  .catch((err)=>{  reject(err);  })  }  })  }  deleteEvent(\_account, eventData){  let findType = this.typeList.filter((type)=>{  if(type.typeName === eventData.title)  return type;  })  findType[0].deleteEvent(\_account, eventData.\_id,calendarSchema);  }  getEventList(){  let eventList = [];  // console.log('here:'+JSON.stringify(this.typeList))  this.typeList.forEach((type)=>{  type.eventList.forEach((event)=>{  eventList.push(event);  })  })  return eventList;  }  }  module.exports = Calendar ; |

**Event.js**

|  |
| --- |
| class Event {  constructor(title, start, end, desc){  this.title = title;  this.start = start;  this.end = end;  this.desc = desc;  }  }  module.exports = Event; |

**CalendarController.js**

|  |
| --- |
| import bodyParser from 'body-parser';  import Account from '../Account';  class CalendarController {  constructor(account){  this.account = account;  }  getTypeList(req, res) {  let calendar = this.account.calendar.getTypeList();  res.send(calendar);  }  getChartInformation(req,res){  let chartInformation = this.account.calendar.getChartInformation(req.query.startDate,req.query.endDate)  res.send(chartInformation)  }  getEventList(req, res) {  let eventList = this.account.calendar.getEventList();  res.send(eventList);  }  }  module.exports = CalendarController; |

**EventController.js**

|  |
| --- |
| import bodyParser from 'body-parser';  class EventController {  constructor(account){  this.account = account;  }  addEvent(req, res){  let event = req.body;  this.account.calendar.addEvent(this.account.account, event)  .then((result) =>{  res.send(result);  })  }  deleteEvent(req, res){  let event = req.body;  this.account.calendar.deleteEvent(this.account.account, event);  res.send('delete Success');  }  }  module.exports = EventController; |

**TypeController.js**

|  |
| --- |
| import bodyParser from'body-parser';  import Type from '../Type';  class TypeController {  constructor(account){  this.account = account;  }  addType (req, res){  let type = req.body;  console.log(this.account);  this.account.calendar.addType(this.account.account, new Type(type.typeName,[]))  res.send('Add Type Success');  }  deleteType (req, res){  let typeName = req.body.typeName;  this.account.calendar.deleteType(this.account.account, typeName);  res.send('Delete Type Success');  }  }  module.exports = TypeController; |

**Account.js**

|  |
| --- |
| import mongoose from 'mongoose';  let connection = mongoose.connect('mongodb://localhost/timelog');  let db = mongoose.connection;  import calendarSchema from './Schemas/calendarSchema';  import Calendar from './Calendar';  import Type from './Type';  class Account {  constructor(account, password){  this.account = account;  this.password = password;  this.createCaldendar().then((result)=>{  this.calendar = result  })  }  createCaldendar(){  return new Promise((resolve, reject) =>{  calendarSchema.find({account:this.account}).then((result)=>{  let typeList = []  result[0].typeList.forEach((type)=>{  typeList.push(new Type( type.typeName, type.eventList));  });  resolve( new Calendar(typeList));  })  })  }  }  module.exports = Account; |

**Type.js**

**(Add new method)**

|  |
| --- |
| import Event from'./Event';  import TimeSetter from './TimeSetter.js';  class Type{  constructor(typeName,eventList){  this.typeName = typeName;  this.eventList = eventList;  }  addEvent(\_account, eventData,calendarSchema){  return new Promise((resolve, reject)=>{  this.eventList.push(new Event(eventData.title, eventData.start,eventData.end,eventData.desc));  calendarSchema.findOneAndUpdate({account:\_account, 'typeList.typeName':this.typeName},{$set: {'typeList.$.eventList':this.eventList}},{ new: true })  .then((result)=>{  let filteredType = result.typeList.filter((type)=>{  if(type.typeName == this.typeName)  return type;  })  this.eventList = filteredType[0].eventList;  resolve(this.eventList);  })  .catch((err)=>{  reject(err);  })  })  }  deleteEvent(\_account, eventID,calendarSchema){  this.eventList = this.eventList.filter((event)=>{  if(\_account === "Hank"){  if(event.\_id !== eventID)  return event;  }  else  if(event.\_id.toString() !== eventID)  return event;  })  calendarSchema.update({account:\_account, 'typeList.typeName':this.typeName},{$set: {'typeList.$.eventList':this.eventList}})  .then((result)=>{  console.log(result);  })  }  getSpecifyPeriodDate(eventDate){    let startDate = new Date(eventDate.start);  let endDate = new Date(eventDate.end);  let timeSetter = new TimeSetter();  let iterativeTimes = timeSetter.timeFormatConvert(eventDate.times);  let firstSpecifyStartTime = timeSetter.specifyDateAndTimeofFirstDay(startDate, startDate);  let firstSpecifyEndTime = timeSetter.specifyDateAndTimeofFirstDay(startDate, endDate);  let PeriodDate = timeSetter.iterativeAddStartAndEndTime(firstSpecifyStartTime, firstSpecifyEndTime,endDate , iterativeTimes);  return PeriodDate;  }  addCycleEvent(\_account, eventData,calendarSchema){  return new Promise((resolve, reject)=>{  let i;  let PeriodDate ;  PeriodDate = this.getSpecifyPeriodDate(eventData);  let cycleDate = {};  for( i=0 ; i <= PeriodDate['start'].length-1; i++ ){  cycleDate.start = PeriodDate['start'][i]  cycleDate.end = PeriodDate['end'][i]  cycleDate.title = eventData['title']  cycleDate.desc = eventData['desc']  this.eventList.push(new Event( cycleDate.title, cycleDate.start, cycleDate.end, cycleDate.desc));  }  calendarSchema.findOneAndUpdate({account:\_account, 'typeList.typeName':this.typeName},{$set: {'typeList.$.eventList':this.eventList}},{ new: true })  .then((result)=>{  let filteredType = result.typeList.filter((type)=>{  if(type.typeName == this.typeName)  return type;  })  this.eventList = filteredType[0].eventList;  resolve(this.eventList);  })  .catch((err)=>{  reject(err);  })  })  }  }  module.exports = Type; |

**TimeSetter.js**

|  |
| --- |
| import moment from 'moment';  class TimeSetter {  constructor(){  // Do not do anything at the current stage;  }  timeFormatConvert(stringOfTime){  let parserStringToNumber ;  if(stringOfTime === '一週')  parserStringToNumber = 7;  else  parserStringToNumber = 1;  return parserStringToNumber;  }  specifyDateAndTimeofFirstDay(specifyDate, specifyTime){  let \_specifyDate = moment(specifyDate);  let \_specifyTime = moment(specifyTime);  return (  new Date(moment().set({  'year': \_specifyDate.get('year'),  'month':\_specifyDate.get('month'),  'date': \_specifyDate.get('date'),  'hours': \_specifyTime.get('hours'),  'minute': \_specifyTime.get('minute'),  'second': \_specifyTime.get('second'),  'millisecond' :\_specifyTime.get('millisecond')  }))  )  }  iterativeAddStartAndEndTime(firstDayStartTime, firstDayEndTime, endDateOfPeriodDate, iterativeTime){  let Times = iterativeTime;  let iteratorDate;  let specifyStartTime = moment(firstDayStartTime);  let specifyEndTime = moment(firstDayEndTime);  let specifyStartPeriodDate = [];  let specifyEndPeriodDate = [];  specifyStartPeriodDate.push(moment(firstDayStartTime).toJSON());  specifyEndPeriodDate.push(moment(firstDayEndTime).toJSON());  while( (iteratorDate = (moment(specifyStartPeriodDate[specifyStartPeriodDate.length-1]).add(Times, 'day'))) < moment(endDateOfPeriodDate)){  let iteratorStartDate = moment().set({  'year': iteratorDate.get('year'),  'month':iteratorDate.get('month'),  'date': iteratorDate.get('date'),  'hours': specifyStartTime.get('hours'),  'minute': specifyStartTime.get('minute'),  'second': specifyStartTime.get('second'),  'millisecond' :specifyStartTime.get('millisecond')  });  let iteratorEndDate = moment().set({  'year': iteratorDate.get('year'),  'month':iteratorDate.get('month'),  'date': iteratorDate.get('date'),  'hours': specifyEndTime.get('hours'),  'minute': specifyEndTime.get('minute'),  'second': specifyEndTime.get('second'),  'millisecond' :specifyEndTime.get('millisecond')  });  specifyStartPeriodDate.push(new Date(iteratorStartDate).toJSON());  specifyEndPeriodDate.push(new Date(iteratorEndDate).toJSON());  }  let specifyPeriodDate = {};  specifyPeriodDate['start'] = specifyStartPeriodDate;  specifyPeriodDate['end'] = specifyEndPeriodDate;  return specifyPeriodDate;  }  }  module.exports = TimeSetter; |

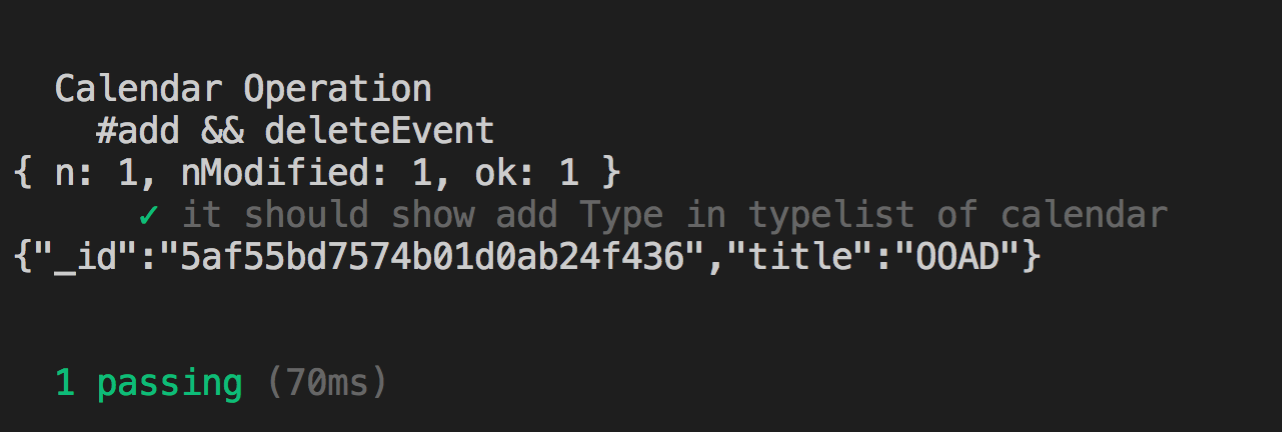
**ChartInformation.js**

|  |
| --- |
| import moment from 'moment'  class ChartInformation {  constructor(){  this.chartInformationList=[];  }  getChartInformation(typeList, startDate, endDate){  let filteredTypeList = this.filterTypeList(typeList,startDate,endDate)  return this.makeChartInformation(filteredTypeList)  }  filterTypeList(typeList, startDate, endDate){  let filteredList=[]  for(let i = 0 ; i<typeList.length;i++){  filteredList.push({  typeName :typeList[i].typeName,  })  filteredList[i].eventList = typeList[i].eventList.slice()  }  filteredList.forEach((type)=>{  type.eventList = type.eventList.filter((event)=>{  if (event.start > new Date(startDate) && event.end < new Date(endDate))  return event  })  })  filteredList = filteredList.filter((type)=>{  if(type.eventList.length > 0)  return type  })  return filteredList  }  makeChartInformation(filteredList){  filteredList.forEach((type)=>{  type.totalSpentHours = 0  type.eventList.forEach((event)=>{  type.totalSpentHours += (event.end - event.start)/ 1000 / 60 / 60  })  })  return filteredList  }  }  module.exports = ChartInformation; |

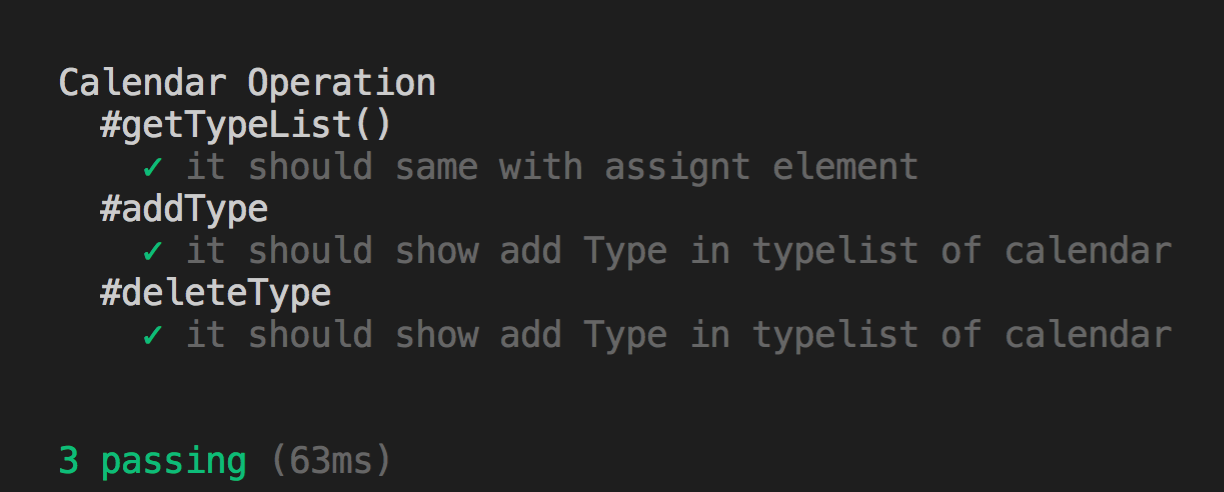
1. **Unit Testing**
   1. Snapshots of testing result

**Event Test**

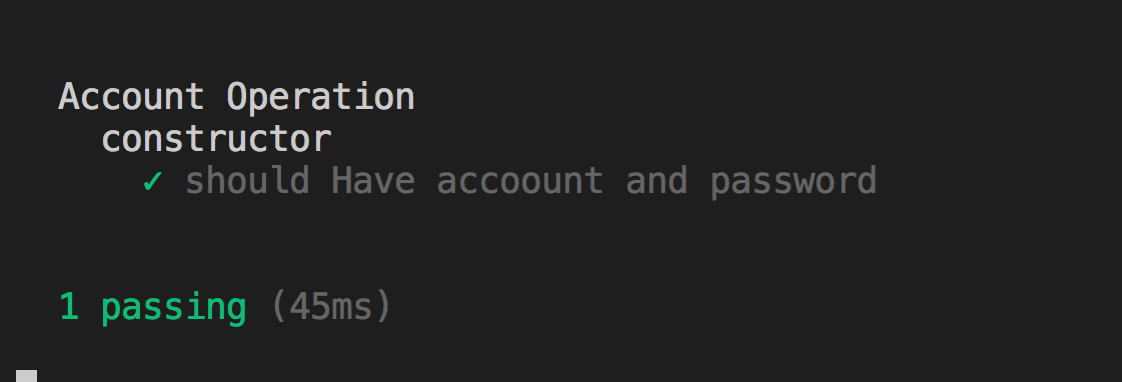
Add And Delete Event In One Test



**Calendar Test**



**Account Test**



* 1. Unit Testing Code Listing

**accountUnitTest.spec.js**

|  |
| --- |
| **let assert = require('assert');**  **let Account = require('../src/server/Account.js');**  **describe('Account Operation', function() {**  **let acc;**  **before(function(){**  **acc = new Account('Hank', '1234');**  **});**  **describe('constructor', function() {**  **it('should Have accoount and password', function() {**  **assert.equal('Hank',acc.account);**  **assert.equal('1234', acc.password);**  **});**  **});**  **});** |

**calendarUnitTest.spec.js**

|  |
| --- |
| **let assert = require('assert');**  **let Account = require('../src/server/Account.js');**  **let Calendar = require('../src/server/Calendar.js');**  **let Type = require('../src/server/Type.js');**  **let Event = require('../src/server/Event.js');**  **describe('Calendar Operation', function() {**  **let acc;**  **let cal;**  **beforeEach(function(){**  **acc = new Account("Hank","1234")**  **let initTypeList = [];**  **cal = new Calendar(initTypeList);**  **});**    **describe('#addType', function(){**  **it('it should show add Type in typelist of calendar', function(){**  **let expectTypeList = [{"eventList": [],"typeName": "STV"}]**  **let type = new Type('STV',[]);**  **cal.addType(acc.account, type);**  **assert.deepEqual(expectTypeList, cal.getTypeList());**  **cal.deleteType(acc.account, type.typeName);**  **})**  **})**    **describe('#getTypeList()', function() {**  **it('it should same with assignt element', function() {**  **let expectTypeList = [];**  **assert.deepEqual(expectTypeList, cal.getTypeList());**  **});**  **});**  **describe('#deleteType', function(){**  **it('it should show add Type in typelist of calendar', function(){**  **let expectTypeList = [{"eventList": [],"typeName": "STV"}]**  **let type = new Type('STV',[]);**  **cal.addType(acc.account, type);**  **cal.deleteType(acc.account, type.typeName);**  **assert.deepEqual([], cal.getTypeList());**  **})**  **})**  **});** |

**eventUnitTest.spec.js**

|  |
| --- |
| **let assert = require('assert');**  **let Account = require('../src/server/Account.js');**  **let Calendar = require('../src/server/Calendar.js');**  **let Type = require('../src/server/Type.js');**  **let Event = require('../src/server/Event.js');**  **describe('Calendar Operation', function() {**  **let acc;**  **let cal;**  **beforeEach(function(){**  **acc = new Account("Hank","1234")**  **let initTypeList = [];**  **cal = new Calendar(initTypeList);**  **let type = new Type('OOAD',[]);**  **cal.addType(acc.account, type);**  **});**  **afterEach(function(){**  **let type = new Type('OOAD',[]);**  **cal.deleteType(acc.account, type.typeName);**  **})**  **describe('#addEvent', function(){**  **it('it should show add Type in typelist of calendar', function(){**  **let typeName = 'OOAD';**  **let type = new Type('OOAD',[]);**  **cal.addType(acc.account, type);**  **let eventData = {**  **"title" : "OOAD",**  **"start" : "2018-05-10T05:33:00.000Z",**  **"end" : "2018-05-10T05:59:00.000Z",**  **"desc" : "coding"**  **};**  **cal.addEvent(acc.account, eventData);**  **let expectEventData = [ {**  **title: 'OOAD',**  **start: '2018-05-10T05:33:00.000Z',**  **end: '2018-05-10T05:59:00.000Z',**  **desc: 'coding'**  **} ];**  **assert.deepEqual(expectEventData, cal.getEventList());**  **// cal.deleteEvent(acc.account,typeName);**  **})**  **})**  **describe('#deleteEvent', function(){**  **it('it should show add Type in typelist of calendar', function(){**  **let eventData = {**  **"title" : "OOAD",**  **"start" : "2018-05-10T05:33:00.000Z",**  **"end" : "2018-05-10T05:59:00.000Z",**  **"desc" : "coding"**  **};**  **let eventData2 = {**  **"title" : "OOAD",**  **"start" : "2018-05-10T05:33:00.000Z",**  **"end" : "2018-05-10T05:59:00.000Z",**  **"desc" : "codingHI"**  **};**  **cal.addEvent(acc.account, eventData2);**  **cal.addEvent(acc.account, eventData).then((result)=>{**  **let deleteData = {**  **\_id:result.typeList[0].eventList[1].\_id,**  **title:result.typeList[0].eventList[1].title,**    **};**  **cal.deleteEvent(acc.account, deleteData);**  **let eventList = cal.getEventList();**  **})**  **})**  **})**  **});** |

1. **Measurement**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Homework** | **Date** | **Time** | **Event** | **Team Members** | **Time(Min)** | **Total Time** |
| 1 | 2018/3/7 | 14:00 - 15:00 | Write the homework 1. | 鍾承翰、楊子冊、吳彥銘 | 60 | 60 |
| 2 | 2018/3/19 | 16:30 - 18:15 | Write the homework 2. | 鍾承翰、楊子冊、吳彥銘 | 105 | 790 |
| 2018/3/19 | 19:45 - 21:20 | Write the homework 2. | 鍾承翰、楊子冊、吳彥銘 | 95 |
| 2018/3/21 | 16:00 - 18:00 | Write the homework 2. | 鍾承翰、楊子冊、吳彥銘 | 120 |
| 2018/3/22 | 17:00 - 18:00 | Write the homework 2. | 鍾承翰、楊子冊、吳彥銘 | 60 |
| 2018/3/22 | 19:00 - 20:50 | Write the homework 2. | 鍾承翰、楊子冊、吳彥銘 | 110 |
| 2018/3/23 | 09:00 - 12:00 | Write the homework 2. | 鍾承翰、楊子冊、吳彥銘 | 180 |
| 2018/3/23 | 16:00 - 18:00 | Write the homework 2. | 鍾承翰、楊子冊、吳彥銘 | 120 |
| 3 | 2018/4/1 | 20:00 - 22:00 | Write the homework 3. | 鍾承翰、楊子冊、吳彥銘 | 120 | 420 |
| 2018/4/2 | 16:00 - 18:00 | Write the homework 3. | 鍾承翰、楊子冊、吳彥銘 | 120 |
| 2018/4/3 | 09:00 - 12:00 | Write the homework 3. | 鍾承翰、楊子冊、吳彥銘 | 180 |
| 4 | 2018/4/13 | 16:00 - 18:00 | Write the homework 4. | 鍾承翰、楊子冊、吳彥銘 | 120 | 826 |
| 2018/4/17 | 15:40 - 18:50 | Write the homework 4. | 鍾承翰、楊子冊、吳彥銘 | 190 |
| 2018/4/18 | 14:30 - 15:00 | Write the homework 4. | 鍾承翰、楊子冊、吳彥銘 | 30 |
| 2018/4/18 | 19:00 - 20:40 | Write the homework 4. | 鍾承翰、楊子冊、吳彥銘 | 100 |
| 2018/4/24 | 18:30 - 20:10 | Write the homework 4. | 鍾承翰、楊子冊、吳彥銘 | 100 |
| 2018/5/2 | 19:00 - 21:15 | Write the homework 4. | 鍾承翰、楊子冊、吳彥銘 | 135 |
| 2018/5/3 | 18:30 - 21:01 | Write the homework 4. | 鍾承翰、楊子冊、吳彥銘 | 151 |
| 5 | 2018/5/9 | 18:30 - 22:00 | Write the homework 5. | 鍾承翰、楊子冊、吳彥銘 | 210 | 930 |
| 2018/5/10 | 17:00 - 23:00 | Write the homework 5. | 鍾承翰、楊子冊、吳彥銘 | 360 |
| 2018/5/11 | 09:00 - 15:00 | Write the homework 5. | 鍾承翰、楊子冊、吳彥銘 | 360 |
| 6 | 2018/5/31 | 09:00 - 15:00 | Write the homework 6. | 鍾承翰、楊子冊、吳彥銘 | 360 | 360 |