SINGAPORE INSTITUTE OF TECHNOLOGY

# ICT 2106 - Software Design

Design Document

**Deadline**
1st April 2019

**Module Professor**
Dr. Fatma Meawad

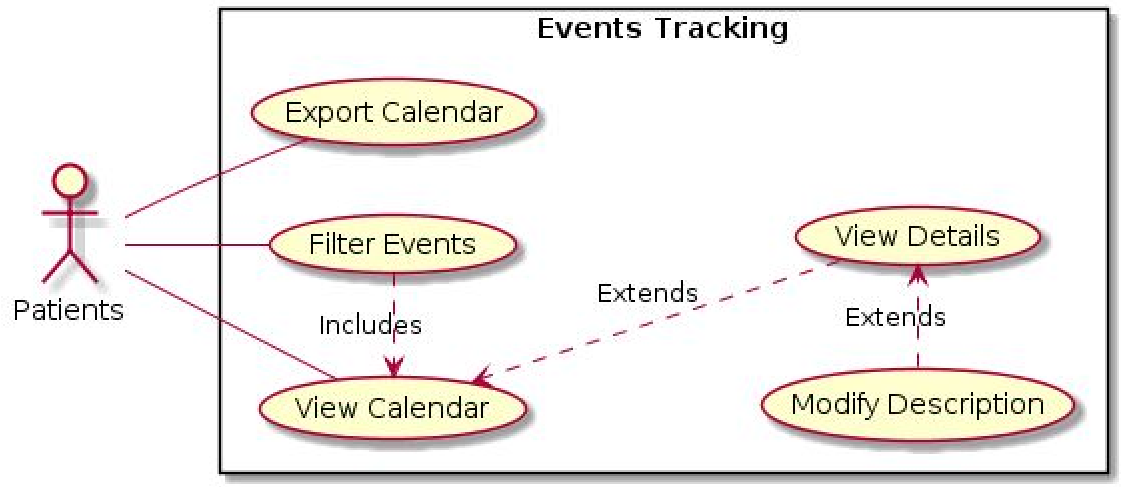
| Matriculation No. | Project Members |
|---|---|
| 1700283 | Leong Wen Qing |
| 1702419 | Ong Xuan |
| 1701591 | Ker Beng Hian |
| 1702327 | Tan Jia Lin Evelyn |
| 1702658 | Alicia Teo Shu Jia |
| 1702251 | Siti Nadhirah Binte Aziz |

## Introduction

The module that the group is undertaking is the task of tracking events.  This feature acts as a summary of the patient's activity, providing the patients with a clearer view of their schedule. For this module, the group will be showing the summary in a form of an calendar view. For the simplicity of this project, the calendar will be accepting only three type of events which are pain episode, medicine intake and doctor appointment. The module will then be called by Pain Diary, Follow-up and Medicine Intake to show their events into the calendar view.

## Use case

This module handles the use cases of a Patient.



The planttext UML for this use case diagram can be found in Appendix A.

1. Patient can *View Calendar*.
2. Patients can *View Details*.
3. Patients can *Modify Description*.
4. Patient can *Filter Events*.
5. Patient can *Export Calendar*.

| **Title** | View Full Calendar |
|---|---|
| **Description** | As a patient, i want to be able to view the full calendar so that i can see my schedule. |
| **Acceptance Criteria** | 1) Patient should be able to see the the full calendar and their event once they are at the calendar page. |

| **Title** | View Individual Event Details |
|---|---|
| **Description** | As a patient, I want to be able to view more event details so that I can know more information about the event that is hidden in the calendar view. |
| **Acceptance Criteria** | 1) Patient should be able to see the details of event once they click on the event on the calendar. |

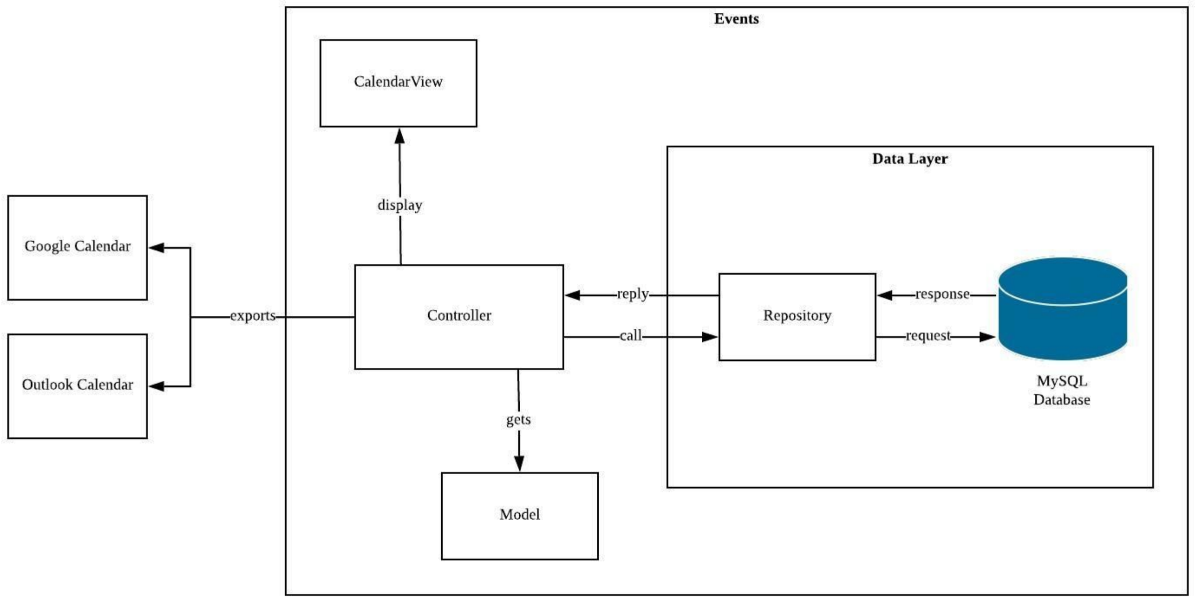
| **Title** | Edit Description Of Selected Event |
|---|---|
| **Description** | As a patient, I want to be able to add and edit description to selected event so that I am able to describe more about the event. |
| **Acceptance Criteria** | 1) Patient should be able to see the details of event once they click on the event on the calendar.<br>2) Patient should be able to add description once they click on the edit button. |

| **Title** | Export To External Calendar |
|---|---|
| **Description** | As a patient, I want to have a choice in integration the calendar with our personal calendar so that I am able to see if there are any conflicts. |
| **Acceptance Criteria** | 1) Patient should able to click on a share icon and see the option menu to choose the calendar that they want to integrate with. |

| **Title** | Filter Event |
|---|---|
| **Description** | As a patient, I want to be able to filter events that I want to see so that it can be viewed in a clearer view. |
| **Acceptance Criteria** | 1) Patient should be able to see only events that they have filtered once they click on the checkboxes. |

## Architecture

### Full architecture of module



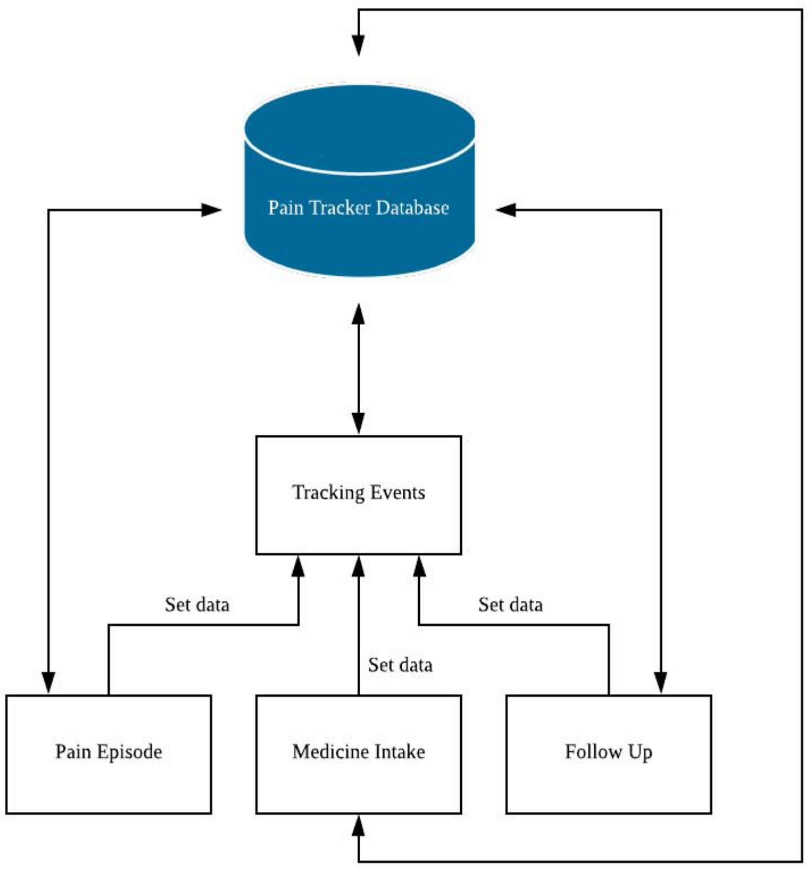
The ASP.NET Core Model-View-Controller(MVC) architecture pattern is used. This pattern separates the application into three main components: Models, Views and Controllers.

Using this pattern, requests from user are routed to the Controller which will be responsible for working with the Model to perform the user requests and retrieve the results. The Controller will choose the View to display to the user and provide it with the Model data it requires.

The Repository will interact directly with MySQL Database to get the data required, and then inject it to the Controller. Services is not created in this case, because the application only gets the data and no logic is needed. Thus, the application only uses Repository.

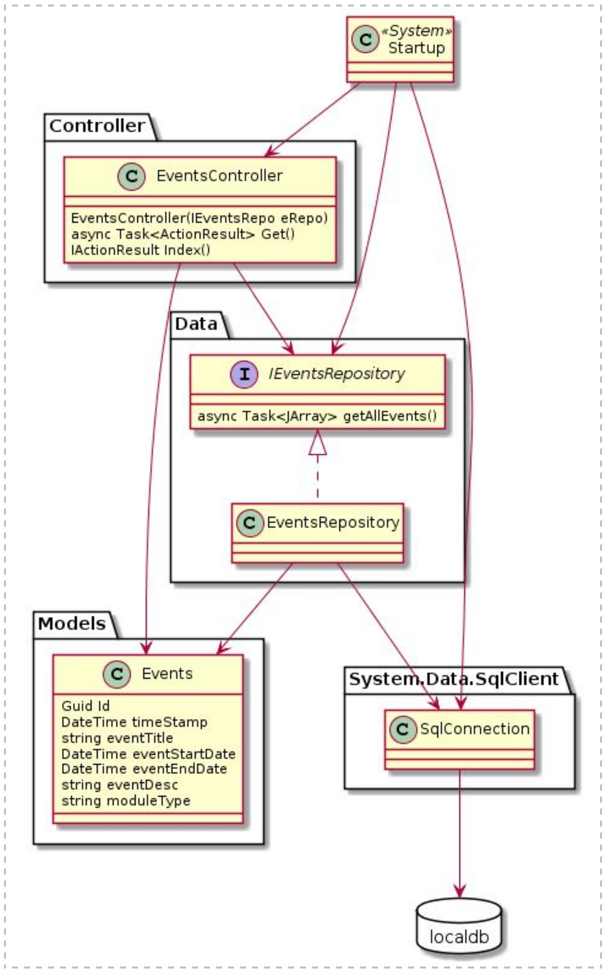
Using this pattern, it is easier to code, debug and test.

## Full Architecture of Project with other modules



The other modules that involves in the implementation of Tracking Event modules are Pain Episode, Medicine Intake and Follow-up. In which, all the modules will interact with the Pain Tracker Database to request for data and modify existing data. Pain Episode, Medicine Intake and Follow Up module will issue the data needed to be displayed in the Patient calendar to the Tracking Events module.

## Class Diagram



The planttext UML for this class diagram can be found in Appendix B.

### Description

The SQL Connection and service is being configured in the Startup.cs class. In order to avoid the controller class from instantiating the context and directly getting information from the database, a repository class, EventRespository.cs is used to access the database context. Hence, the controller class, EventController.cs will instantiate the repository class, handle any incoming request and retrieves the necessary model data from Event.cs model class, and return appropriate responses.

## Package/Layer

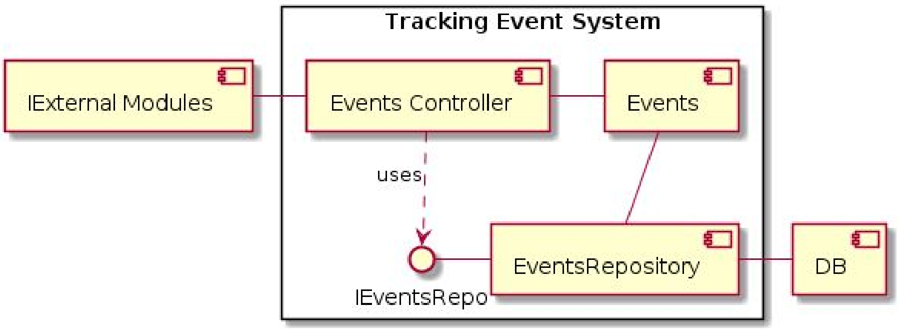
The module is divided into 4 different layers - Model, Controller, Data and SqlConnection.

- The package Model contains classes that holds data of the application.
- The package Controller contains classes will handle incoming request to the application, retrieve model data, and specify view template that return a response to the client.
- The package Data contains classes that will interact with the data in database
- The package SQLConnection will instantiate the connection with database.

## Design Pattern

We have chosen to use dependency injection as one of our design patterns which allows other modules to call our module. It allows us to develop a loosely coupled code and at the same time it makes the code easily maintainable. Using transient service, it allows us to ignore problems like multithreading and memory leaks as  we know the service has a short life.

# Component Diagram



The planttext UML for component diagram can be found in Appendix C.

The IExternalModules component refers to interface class of Pain Episode, Follow-up and Medicine Intake which will interact with the Tracking Event System. The IExternalModules component will request data through Event Controller component. The EventsController component requires data from the Events component, and uses the EventRepository through IEventRepo interface. The EventRespository is used to request data from the DB component, which refers to the database of the system. The EventRespository will then provides the data requested to Events component.

## Design Concerns and Challenges

### Challenges

During the development of this project, the group has faced difficulties in integrating the previous working codes with the github repository. Many changes needs to be done for the codes to be dependent and loosely coupled thus, the group has trouble breaking them into the different packages at first. However, after referring to the codes in the lecture, the group managed to find ways to solve the issue.

Designing the class diagram was not a breeze for the group as a lot of new rules and dependency were introduced for the second half of the project. We had to redesign the structure of the application while keeping the functionality there.

As we are working on an individual module of the project, collaboration with other modules is important. We have to consider the ease of implementation for other modules whenever we are designing our classes. Our methods has to be easily accessible from other modules at the same time keeping variables private to only our classes.

### Future Development

The calendar now only allows user to export to external API such as google and outlook calendar. However, for better usability, we can allow import of personal calendar into the application instead. This will allows user to be alerted of any conflicts when events are schedule into the calendar such as doctor appointment. With the importation function, user can be assured that no conflicts of personal and medication schedules are found.

Another way the calendar can be further improved is to make it shareable to selected personal. By doing so, family members or closed relatives can ensure that the patients follows the calendar plans such as doctor appointment. This feature will increase the feasibility of the calendar and the inter-personal touch to it.

Icon can be customisation by user to increase the personality and clarity of the calendar. Instead of the fixed icon given by the system in accordance to the type of event, patients can select their own icons and colors to personalised the calendar.

For now, the calendar can only be viewed on the website. However, for the increase usage of calendar, the module can be further developed into a working application for better easability.

## Task Assignment

| Task | | Team Member Assigned |
|---|---|---|
| 1. | View Full Calendar | Tan Jia Lin Evelyn |
| 2. | View Individual Event Details | Ong Xuan |
| 3. | Edit Description Of Selected Event | Siti Nadhirah & Ker Beng Hian |
| 4. | Export To External Calendar | Alicia Teo Shu Jia |
| 5. | Filter Event | Leong Wen Qing |

## Reflections

### Team Reflection

As the project move forward, the team realised that the project is not on par with the SOLID framework. Although it was a working project, there are violations to the principles. For example, in the previous project, the team was missing the service class where the logic should be placed in.

The group has understood the importance of the SOLID principles and tune the project to suit the requirements. SOLID helps in reducing the tight coupling between the different classes and increase the reusability. Using the SOLID principle as a guideline, it helps each member of the group to know more about the right way to create the various classes in the software design.

### Leong Wen Qing

In the first part of the project I worked on the features that was required in the project requirements. The features are working as expected but the, however, I found that the methods that are troublesome to modify due to changes.

After the second part of the module, I've learnt more about the SOLID principles which allows me to create files that are easily reusable as well as easy to modify to change or add new features.

### Ong Xuan

The project has taught me the importance of having a proper naming conventions and the designing of the project. Due to these importance, having learnt more about SOLID has open up many new ways to reduce codes smells and increase the quality of the codes.

Using this project as an example, from the initial submission to the final submission, there is a great difference in the way that the codes are placed and called. The final submission follows the SOLID principles and by doing the project, i can better understand what the codes does. Other than that, having proper SOLID codings allows the project to be reusable and extendable.

### Ker Beng Hian

This project has taught me about the various principles and design that helps to produce a quality project. I have learnt about the differences between different patterns, such as creational patterns and structural patterns. I have learnt the use of dependency injection, which injects a service, which also allows the program to be loosely coupled.

I have also learnt about SOLID principles and the reason behind why the project has to fulfill the principles. The project has to confront to SOLID principles to ensure that the project will be scalable and the functions can be reusable when other functions needs to be added into the project.

### Tan Jia Lin Evelyn

By doing this project, I am able to learn the importance about the implementation and design aspects with the proper coding style. The most significant takeaway from the project is the SOLID principles where i am able to understand how to apply these principles into our project properly through the project, to be able to code in a better quality coding pattern.

With the application of SOLID principles, we will be able to improve and present design patterns in a more understandable, flexible and maintainable way. It also allows the next programmer taking over the project to be able to understand the codes and allows existing functions to be reusable more easily.

### Alicia Teo Shu Jia

From this project, I am able to learn more about the SOLID principle and how to implement them into the application. I now have a clearer concept for developing software application in a proper way to avoid bad design.

It is important to design the software such that no high-level modules should depend on any low-level modules. In addition, a service should be created to for the business logic and inject it into the controller. With these knowledge, I can now write better quality code.

### Siti Nadhirah Binte Aziz

This project has shown me that there are proper naming convention and how a bad code will make the project hard to maintain. I learned what are the different types of SOLID principle there are, and how to apply it.

I have also learn about a new website, called planttext.com, that helps to draw out the different diagrams to show how the team project works. It also teach me that for different framework, there are different way to draw the diagrams out.

## References and Glossary

| Glossary | Description |
|---|---|
| External Modules | Modules that provides the event data. This includes follow-up module, medicine-intake module and pain diary module. |
| Follow-up | One of the modules in the pain tracking application. The module provides follow-up events that are initiated by the doctor. |
| Medicine-Intake | One of the modules in the pain tracking application. The module provides events that notify patient when to take medicine. |
| Pain Diary | One of the modules in the pain tracking application. The module provide events that are entered by the patient. |
| Patient’s activity | An overview of all the event that belong to the patient. |
| Summary | An overview of all the patient’s activity that will be display in calendar. |
| Tracking Events | One of the modules in the pain tracking application. The module get all events from the other module and display it in the application’s calendar. |

# Appendix

## Appendix A

| Use Case |
|---|
| @startuml<br>left to right direction<br>:Patients:<br>rectangle "Events Tracking" {<br>:Patients: -right- (Export Calendar)<br>Patients -- (Filter Events)<br>(Filter Events) .right.> (View Calendar) : Includes<br>Patients -- (View Calendar)<br>(View Calendar) <.. (View Details) : Extends<br>(Modify Description) .right.> (View Details) : Extends<br>}<br>@enduml |

## Appendix B

**Class Diagram**

```
@startuml
package "Models"{
class Events {
Guid Id
DateTime timeStamp
string eventTitle
DateTime eventStartDate
DateTime eventEndDate
string eventDesc
string moduleType
}
}

package "Controller"{
class EventsController  {
  EventsController(IEventsRepo eRepo)
  async Task<ActionResult> Get()
  IActionResult Index()
}
}

package "Data"{
class EventsRepository  {
}

interface IEventsRepository  {
async Task<JArray> getAllEvents()
}
}

package "System.Data.SqlClient"{
class SqlConnection  {
}
}

class Startup <<System>>  {
}

package localdb <<Database>>{
}

EventsController --> Events
EventsRepository --> Events
EventsController --> IEventsRepository
IEventsRepository <|.. EventsRepository
EventsRepository --> SqlConnection
Startup --> EventsController
Startup --> IEventsRepository
Startup --> SqlConnection
SqlConnection --> localdb
@enduml
```

Appendix C

## Component Diagram

```
@startuml
skinparam componentStyle uml2

component DB
[DB]

rectangle "Tracking Event System" {
	DB -left- [EventsRepository]
	[Events Controller] - [Events]
	[Events] - [EventsRepository]
	[Events Controller] ..> ()IEventsRepo :uses
	IEventsRepo -left- [EventsRepository]
}
[Events Controller] -left- [IExternal Modules]

@enduml
```