Design and Technology manifesto

PHP with the framework CodeIgniter was chosen because of member familiarity and recommendation. Other alternatives were using Django or some similar technology, it would however been new waters for every member. This resulted in us deciding on CodeIgniter for PHP. The architecture of the website uses the MVC model because it is enforced by CodeIgniter. It also allows different responsibilities for members so the work is divided into almost separate entities. The data model of our project is given in next page.

We used FullCalender and Qtip2 to help us in this project. FullCalendar is a calendar view developed in jQuery which provide API to drag, resize events and also it also provide multi-views such as month, week and day views. We chose FullCalendar because our project is based on event booking and we needed a Calendar to show all the events. We had two options in order to show a calendar view, which are Google calendar view which is quite popular and FullCalendar. We chose FullCalendar because it gave us a lot of room to customize it into our intention. We also don't want to share our product champion's information with Google in order to protect their privacy. By using FullCalendar, we can control all the data and customization.

We also used Qtip2 to show the event's description when the user hover over the event. We felt this important because it gives an overview about the event and user doesn't need to go into edit event page just to see the details. This also will help admin to take quick decision on confirming the event. We chose Qtip2 instead of other tooltip tools because Qtip2 is closely developed with FullCalendar and it is easy to implement with FullCalendar.

We made several design and UI decisions in this Sprint. We decided to give admin easier way to confirm the events. This is because Storefront has few clients and most of the time the admin will prefer to confirm the events. We did this by letting admin to click on the event and the event will be confirmed. We also color-code the events according to their status. We gave green to confirmed event, blue for tentative events and red for rejected events. This will make easier for admin and client to see their event's status just by a glance.

```
Project
├── Models
│   ├── Rooms — This model handles database get and set for room table in database
│   ├── Client — This model handles database get and set for client table in database
│   ├── User — This model handles database get and set for user table in database
│   └── Booking — This model handles database get and set for booking table in database and set the format of time and date properly for booking event
├── Views
│   ├── main — Main has the view of claendar and link to all the other views
│   ├── booking — Two forms in this view, one is to create new event, other one to edit events
│   └── account — consist forms of login, edit clients, edit users, create client and create users
└── Controllers
    ├── Account
    │   ├── Get the values from views and parse it correctly in order to call model functions to create or edit users
    │   ├── save user information upon login to access later for certain priviledged action
    │   └── prevent direct access to views
    └── main
        ├── controls all the booking view and main views
        └── handles user interface with calendar and saves the changes the user making on the calendar
```