Bilkent University

Department of Computer Engineering

Object-Oriented Software Engineering Project

CS 319 Project: Tempo

Design Report

Project Group: 1.G

Member Names: Mert Saraç, A. A. M. Jubaeid Hasan Chowdhury, Burak Erkılıç, Kaan Kıranbay

Course Instructor: Eray Tüzün

Progress Report
March 03, 2018

Table of Contents 1

Analysis Report

CS 319 Project : Tempo

1. Introduction

In the first section of the design report, we will start with the explaining our purpose of system. Then we will mention our design goals in detail. We have explained some keywords, acronyms, and abbreviations about the system too. These are the main things that are considered while making the rest of the report.

1.1 Purpose of the System

Tempo is an application that aims to ease the life of the users by helping people to choose time for any kind of events for themselves or between their friends or coworkers, and by reminding people that an event that they should do or they want to do is approaching . The user interface is designed to maximize users' desire to use the application by its' crisp and smooth visuals and its' easy usage. The application will use internet to get information about events and users' friends. These feature makes the application kind of a different from other calendar applications which makes this application more enjoyable to use. Therefore, the purpose of the system is to deliver users a different experience for arranging events.

1.2 Design Goals

Main goals of this application's designs are simple: ease of use and satisfactory visuals. To accomplish these we have to trade some other important design components. In this part, we will talk about what our design goals are, what they mean to us, and the trade-offs.

End User Criteria:

Ease of Use: It can be advocated that people do not want to waste too much time to organize their daily routine so, easiness of the system is one of the core features. To makes organizing process faster, Tempo provides some mouse actions such as moving events to another time, creating new event by dragging specific time period in the calendar. In addition, having one main screen that contains many useful lists like To-Do-List and Friends List keeps all of the core functionality in the home page and allow quick access.

Ease of Learning: Since our program is designed for diverse group of age, ease of learning is important. In this way, our design is similar with the popular programs that users are used to due. Even though, in case of they are not used to some features, there is a help button to explain all features of the program such as creating events, explanations of events etc.

Satisfactory Visuals: To attract and encourage users, the program has many graphical features. All events have a color and it is changeable and various stickers can be attached to describe a event in a better way. Also, users can set a background wallpaper for calendar. Finally, night mode is available for users who wants to use the program at night so, brightness of the program does not hurt their eyes by changing default color scheme from light to dark.

Increased Productivity: As increasing productivity is one of our main goal, a lot of the designs are based on that. One of the core feature is smart event in which the user can invite all of his friends for group events such as birthday party without individually contacting them. This allows for arrangement of events for large groups without having to waste any time. Also notifications before any event aids the user to prepare for the event in early and attend a event which he otherwise might forget. Also to-do-list allows the user to group subtasks such as shopping list that can aid the user to make sure he has completed a main task.

Maintenance Criteria:

Reliability: System should be purified from serious bugs and system crash problems. In order to achieve this, many testings will be done and while the implementation of program, all cases are approached carefully. In addition, using Singleton prevents some potential bugs in the system. For example, it is not possible to create more than one instance of Network class. However, if the system will crash, system send a system crash report to inform developers. Another problem is disconnection from internet and by having an offline mode, this problem is prevented partially. User cannot finalize a SmartEvent until becoming online. (Singleton)

Modifiability: To maintain a system, adding new features and updates are important to not lose users' interest and object oriented programming enables to easily change and add features without any bug. In addition, our program is open system for adding new features like chat between users.

Adaptability: In order to use the system in various platforms, the system is implemented by Java which is provides an easy way to implement for cross-platforms. The system will work without any problem in all JRE installed platforms. It is the main reason that we preferred Java in our implementation.

Performance Criteria:

Efficiency: Tempo's system is a responsive system which means it should be fast when users want to use or change anything in the system. Therefore, the program won't use much memory. The system's memory usage will be less than 10 megabytes. The system is able to do this by not using big pictures such as high quality pictures. The responsive system is helpful while using the database too because the system will be getting information from the users, sending that information to database and saving it, and if needed the information will be received from the database. These informations will not have need large memory spaces to be stored. Therefore, the responsive system and efficient system will make this program's internet usage as low as possible like 1 Kbps download and upload.

Response Time: The system of Tempo uses little memory and internet to run the program. This helps program to run fast and have small latency. Some core features like adding a personal event or changing the time of the events or registration to the system will be done in at worst 100 milliseconds. However, some things like adding a friend or adding a Smart Event and inviting people to it will be kind of a slow compared to other features like adding a personal event to the calendar; because when adding people or adding smart events, the system has to send information to database and from database the system will use that information to send invites to the invited people. However, these won't be that slow like 2 seconds but it will take the twice or the 3 times time.

Connection Time: As already mentioned on the previous part, the system uses little memory and internet. This helps the system to connect to database and gather information with low latency. However, there is 2 more like connecting for the first time and connecting after a disconnect.These two will be treated together as just connecting to the system. Connection time to the system will be at worst 500 milliseconds.

Trade-Offs:

Usability vs. Functionality: As a team, we concluded that we should have a basic system for the sake of users because the users will be everyone. Therefore, the system should not be complex to use. To accomplish that we have to use not complex and not much functions. This will help users to learn and remember the functionalities of the system easily. All of these will make users enjoy using the system.

Performance vs. Memory: The system is planned to be efficient and responsive which means high performance and low latency. To accomplish those we have to sacrifice memory. Even though we are not using that much memory, it can be said that we sacrificed memory for the system's performance.

Efficiency vs. Reusability: In the system, one of the main goal is efficiency since reusability is not one of the concerns of us. In our design, none of our classes will be used for another project so, our design will be more functional by having simple implementation.

1.3 Definitions, Acronyms, and Abbreviations

Since the project has many features, there are some terms that are should be explained:

Smart Scheduling: This feature makes people be able to use their time more efficiently with their friends or co-workers by showing the common time among people to the users.

Event: This feature is the core feature of Tempo. An event is simply an activity that can be whether with your friends (that is called SmartEvent) or without them (that is called Personal Event). In addition, it has some other categories like timeless event, timing event, temporary event and permanent event so, Tempo enables users to combine these categories for describing their event in the best way.

Smart Event: This feature is an event that is created by one of the users' requests to other users who are attending this meeting. For this, when one user is scheduling, he/she can see who is available during this period so he/she can easily change the time period for other individuals. When decided on the time, the user request for Smart Event and then, other participants are notified for this meeting, its explanation, participants, time period. Other participants can whether accept this request or reject. When one participant accepts it, the system waits for other participants response. If all of them accept it, this Smart Event is added to all participants schedule. If any participant does not accept it, system inform the one who creates the event for who did not accept and asks "Do you want to still arrange this event?" without the one who did not accept.

Personal Event: This feature is an event that user's friend list cannot see what user is doing on this time period but they can see he/she is not available during this time period.

Timeless Event: This event type is that is can happen anytime in a day. In simple words, there is no specific time period to do this event like drinking water.

Timing Event: This event type has specific time allocated time for doing this event. For schedule this, user use the weekly-time table.

Temporary Event: This event type is one-time event unless user specifies it will repeat.

Permanent Event: This event type is repeated so when user adds this event to his / her schedule, it will remain until user decides to this event will not be repeated anymore. For this feature, user can adjust how often this event will repeat like every day, every other day, weekly etc.

Abbreviations:

- **JRE** : Java Runtime Environment. It’s a software package that contains what is required to run a Java program.[1] Therefore, JRE needs t o be installed on your computer in order to run Java applications.[2]

2. Software Architecture

In this section, the decomposition, hard/software mapping, persistent data management, access control and security, and boundary conditions will be explained in detail. Main goal is here to reduce the coupling between the different subsystems of the system, while increasing the cohesion of the subsystem components.

2.1 Subsystem Decomposition

Our design contains 6 main components that are designed for various features of the system. These are Authorization, Data Management, Event management, Profile Management, Dynamic Notification Management and SmartEvent Management due to it contains many features of other components as it is shown in Figure 1.

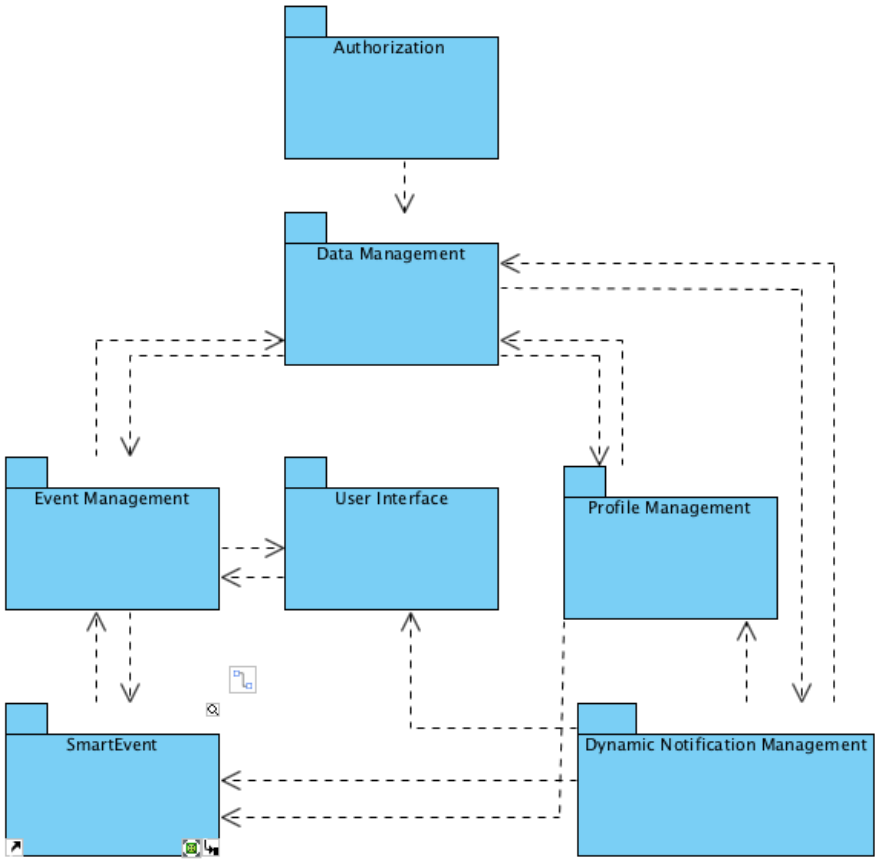

Figure 1 - Representation of Subsystem Decomposition

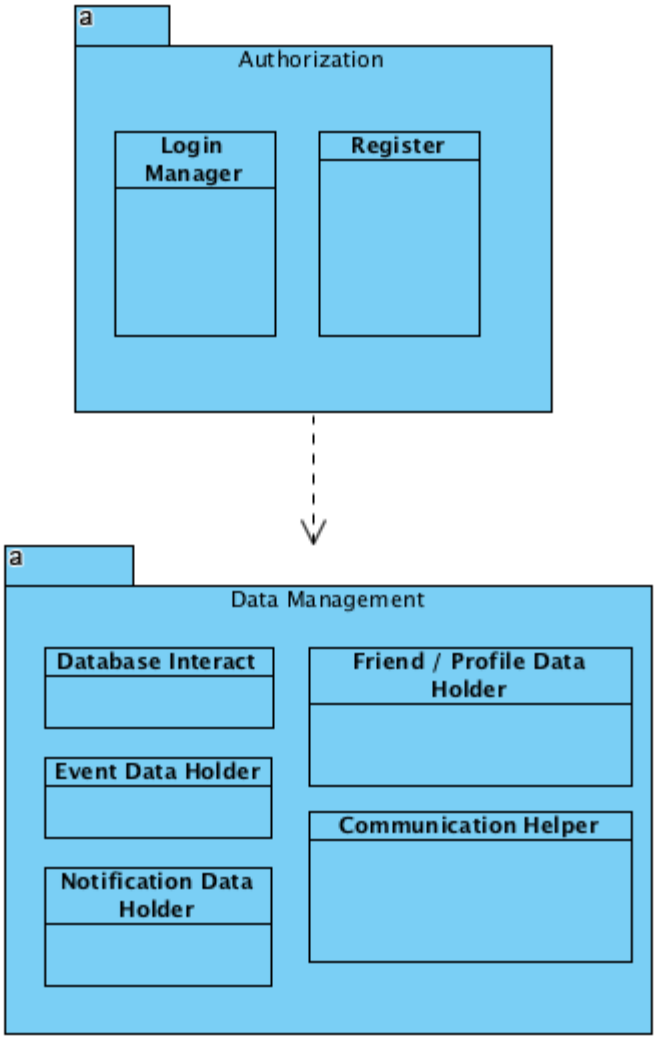

Figure 2 - Layer 1

The authorization subsystem collects user info which is then checked against a central dataholder in the database to match the infos and allow the user to access his own profile.

The data management section is responsible for storing all the model objects and interacting with the database. It contains 3 data holder classes for storing event objects, profile objects and notification objects. The Communication Helper class allows easy communication between holder objects and the database. The Database Interac class is responsible for establishing networking with database.

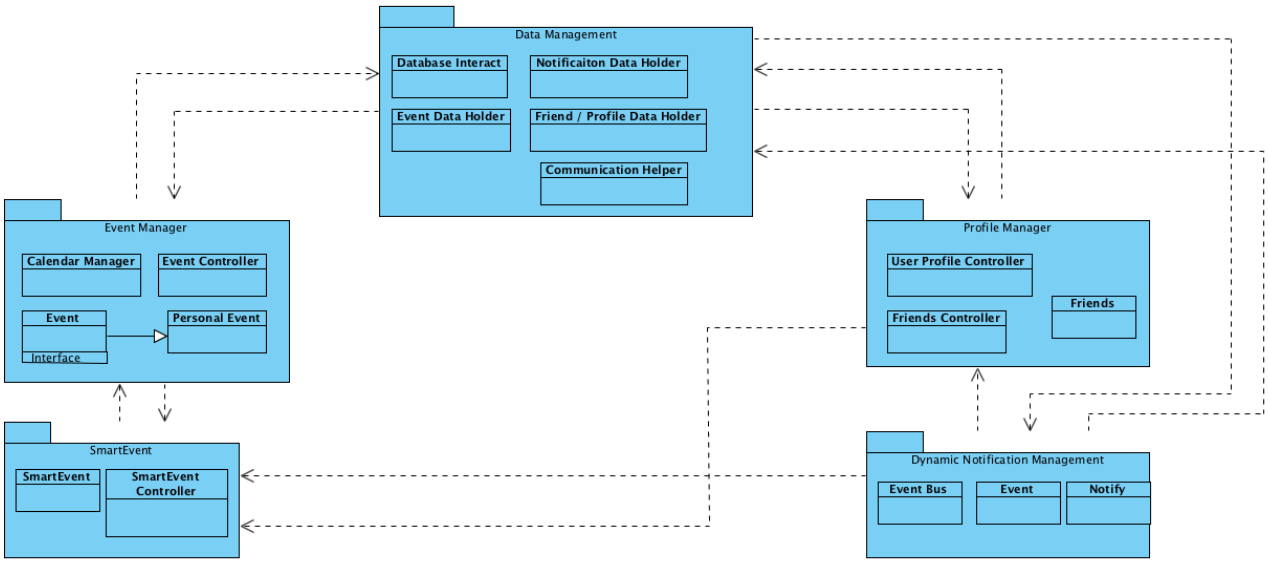

Figure 3 - Layer 2

The SmartEvent manager is a subsubsystem to event manager which is responsible for creating and controlling Smart Events. However, the displaying of this smart events is still done by calendar manager. The profile management subsystem contains User Profile Controller which controls user profile information and Friends Controller controls friend lists and creates new friends.

The system contains a subsystem named Dynamic Notification Management which contains the Event Bus which has an active connection with the database and communicates with it about friends requests and Smart Event requests of users. If any friend request gets accepted then Event Bus gives a pop-up notification and sends message to Friends Controller to add that user to friend list. If any Smart Event request gets accepted by the users then Event Bus gives a pop-up notification and sends message to Notify class to notify the users that the event is added to their calendars.

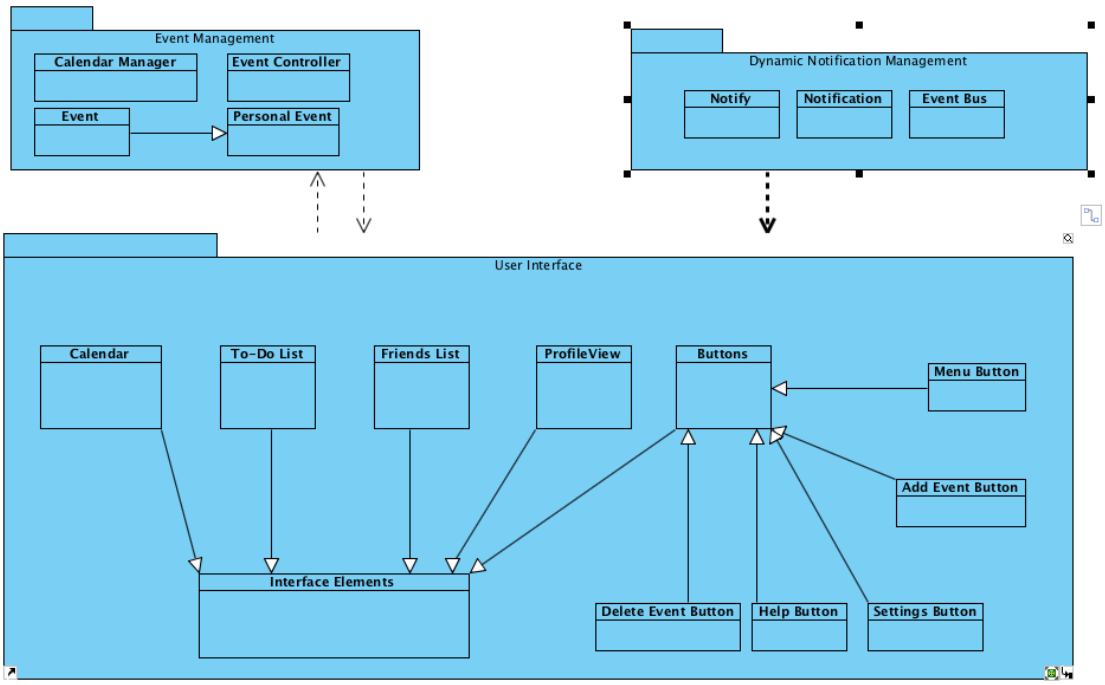

Figure 4 - Layer 3

The user interface subsystem contains a javaFX API calendar and GUI material that are used to display events and to-do-list. The event management subsystem is the controller section for creating and displaying all types of personal events. This section contains Calendar Manager as a controller to Interface Elements in User Interface and Event Controller to control personal event objects and personal event model object.

2.2 Hardware / Software Mapping

Since our system is developed by Java, it needs Java Runtime Environment to be executed. For hardware requirements, the system needs a keyboard and a mouse to be controlled by user and also for interaction between them. For graphical part, Java graphics do not need too much GPU and it is compatible for majority of the personal computer systems.

2.3 Persistent Data Management

Most of the data is stored in the database. However, if the user is already logged in by using internet connection from a device, the user will be able to use the application's some of features like adding personal event offline. To acquire this feature, the system uses data management subsystem to store some of the data in device memory like saving events that are already in the calendar.

2.4 Access Control and Security

User needs to register or login for the usage of the system. By asking two times for users' new password eliminates the problem of creating a password that a user does not know. Also a user needs to enter name, surname and valid email address to enroll to the system correctly. For login part, unique username and password are needed to access the profile and personal calendar. Users' passwords will be matched by md5 encoding to be secured.

2.5 Boundary Conditions

Initialization

Tempo is a click and run application for the personal computers that have java runtime environment. Therefore, there won't be a install file for Tempo. Tempe is a application that is an executable .jar file.

Disconnection

Tempo is a system that works with internet connection. In case of disconnection, many features of the system become not usable due to not connecting with the database.

Termination

Tempo can be terminated or closed by clicking the exit button on the user interface all the times. When a user clicks exit(X) button, the system will disconnect itself from the database. However, this termination will not log the user out of the system. The user will be able to use the system if there is no internet connection as an offline user.

Error

The system will give an error if the file or the saved data in the database are corrupted and will delete its contents.

3. Subsystem Services

In this section, the design patterns, the detailed design of the class diagram, and the explanation of the classes can be seen.

3.1 Design Patterns

Model–view–controller (MVC): This design pattern used for developing user interfaces. It divides an application into three parts. This is done to separate internal representations of data from the ways data is presented to and accepted from the user. This design pattern is used for user interface aspect of this project. Using this architecture style made our subsystems more dependent on each other and made the system more efficient. Therefore, using MVC is a good choice for this system.

Publish & Subscribe Pattern: Publish–subscribe is a messaging pattern where senders of messages, called publishers, do not program the messages to be sent directly to specific receiver classes, called subscribers. Instead of this, the system connect both sides by instant cycles. This design pattern is using for developing an Event Bus object which will notify other classes in order to refresh subsystem's inputs and better communication between user and network.

3.2 Detailed Class Diagram

In Figure 5, Detailed Class Diagram is shown to expand the understanding of the basic functions ,the interactions among them and how components work one by one.

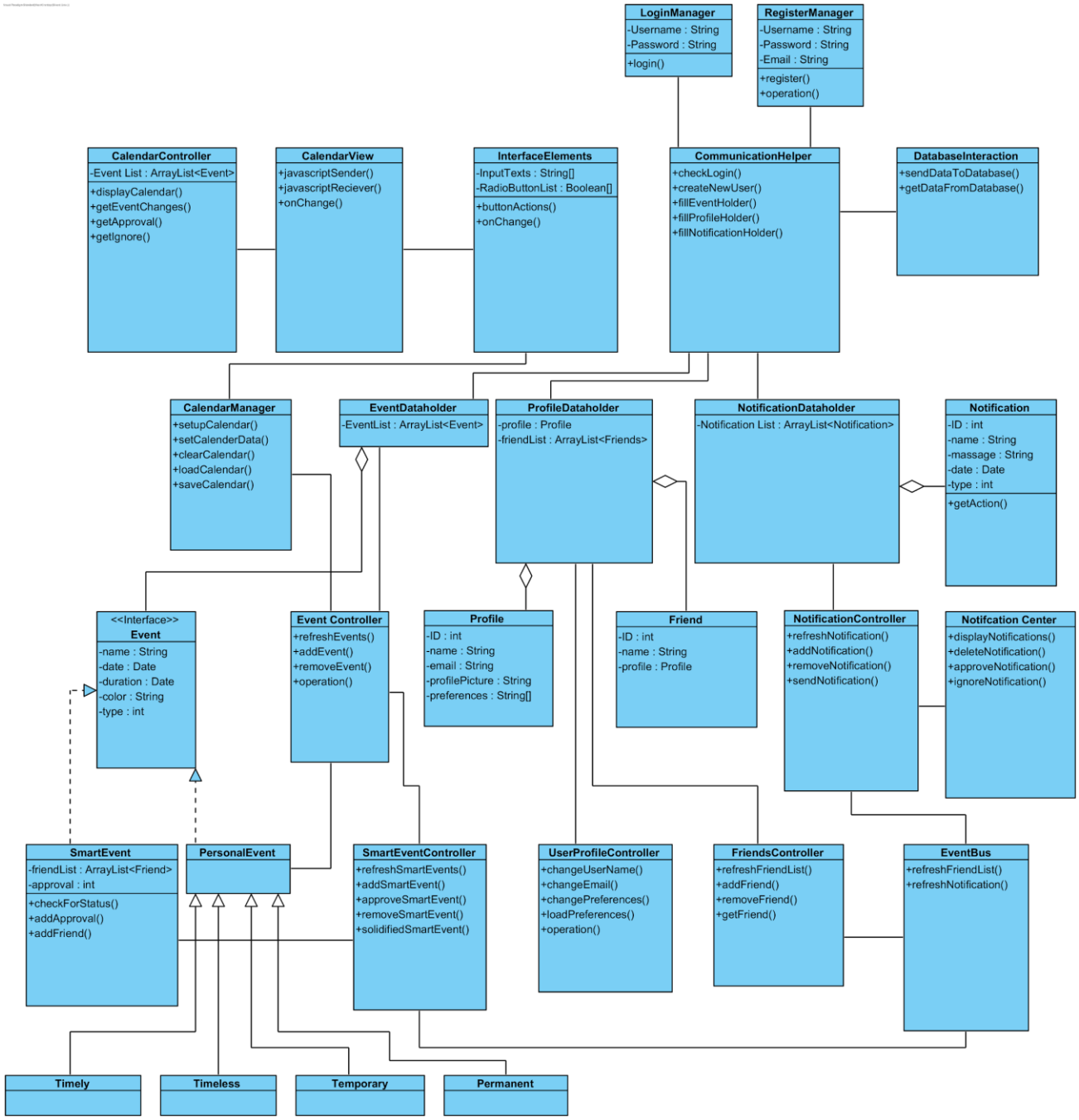

Figure 5 - Detailed Class Diagram

3.3 Class Interfaces

3.3.1 Authorization Subsystem Interface

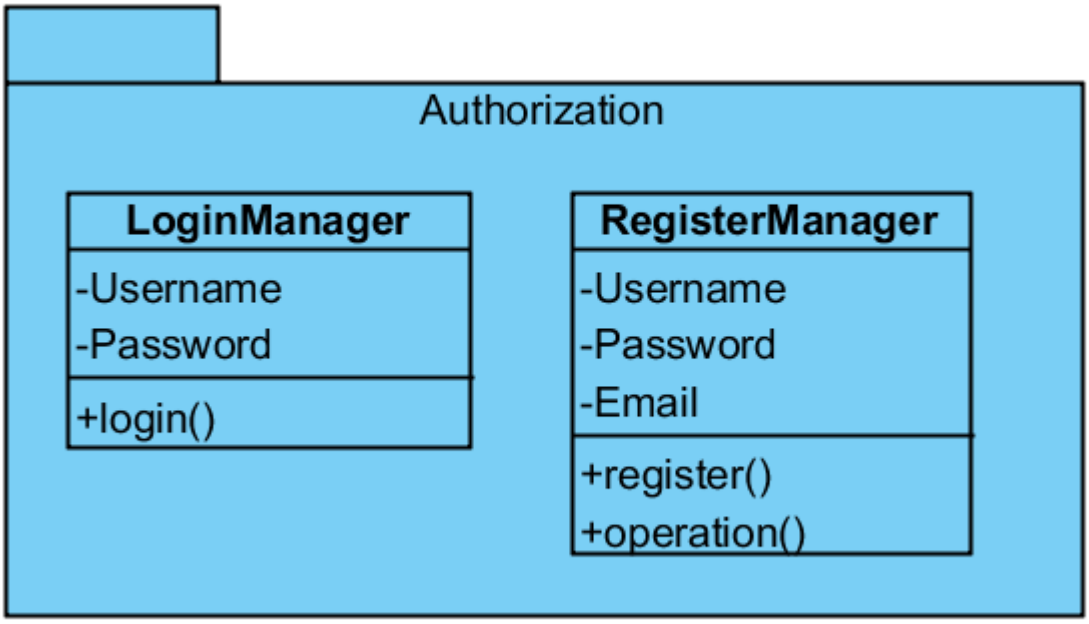

Figure 6 - Package Diagram of Authorization Subsystem

Authorization Subsystem helps the system to connect to the database and login or register the user.

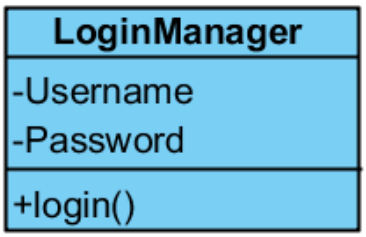

Figure 7 - LoginManager Class

To login, user needs to enter unique username and unique password to access the profile. Access is provided by login method.

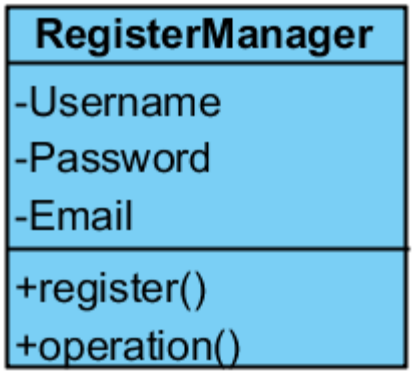

Figure 8 - RegisterManager Class

To register, user needs to give informations which are username, password and email. Username should not be taken by another user to register successfully. Informations are saving into database by register method.

3.3.2 Data Management Subsystem Interface

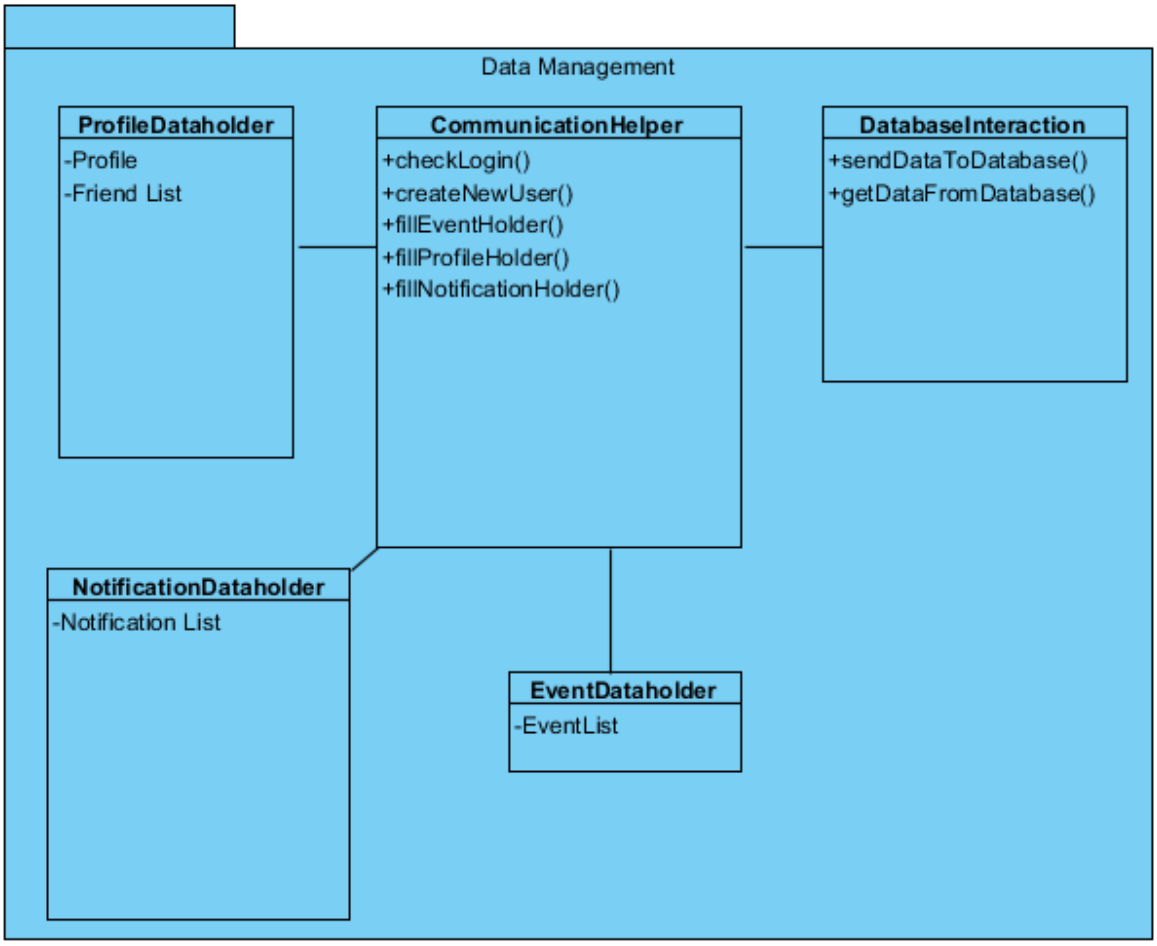

Figure 9 - Package Diagram of Data Management Subsystem

The Data Management system holds the data in device memory using 3 holder classes. Other 2 classes helps the system to communicate with the other classes of the system and the database.

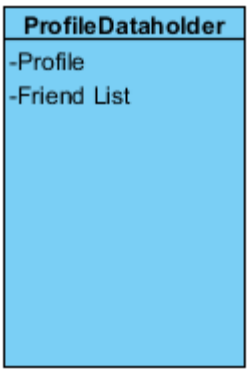

Figure 10 - ProfileDataHolder Class

This class holds the information about user's profile and their friend list.

Figure 11 - NotificationDataHolder Class

NotificationDataHolder class holds the notification objects and keeps them as lists.

Figure 13 - EventDataHolder Class

EventDataHolder class keeps track of every event that the user has in his calendar as a list.

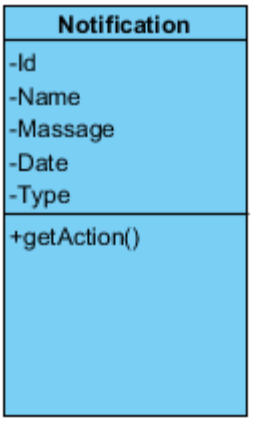

Figure 12 - Notification Class

Notification class helps system to gather information about who will receive the information, when, and the massage that it will show to the sent user.

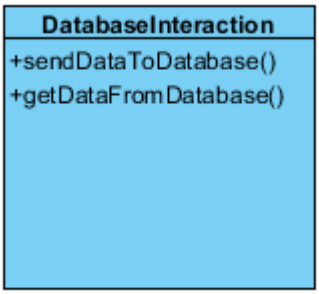

Figure 14 - DatabaseInteraction Class

DatabaseInteraction class helps the system to send and get data from the database.

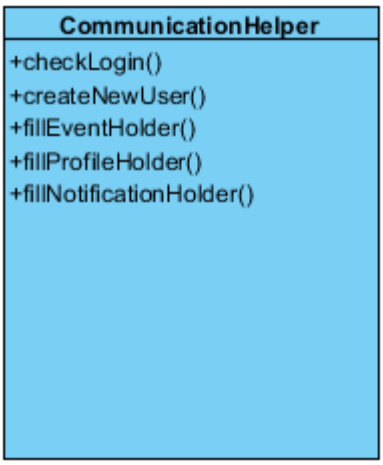

Figure 15 - CommunicationHelper Class

CommunicationHelper class as can be understood is a helper class to the system. It helps the system to check if the user is logged in, if the user created a new account and fills the new data with the help of DatabaseInteraction class, and other classes in the subsystem.

3.3.3 Event Management Subsystem Interface

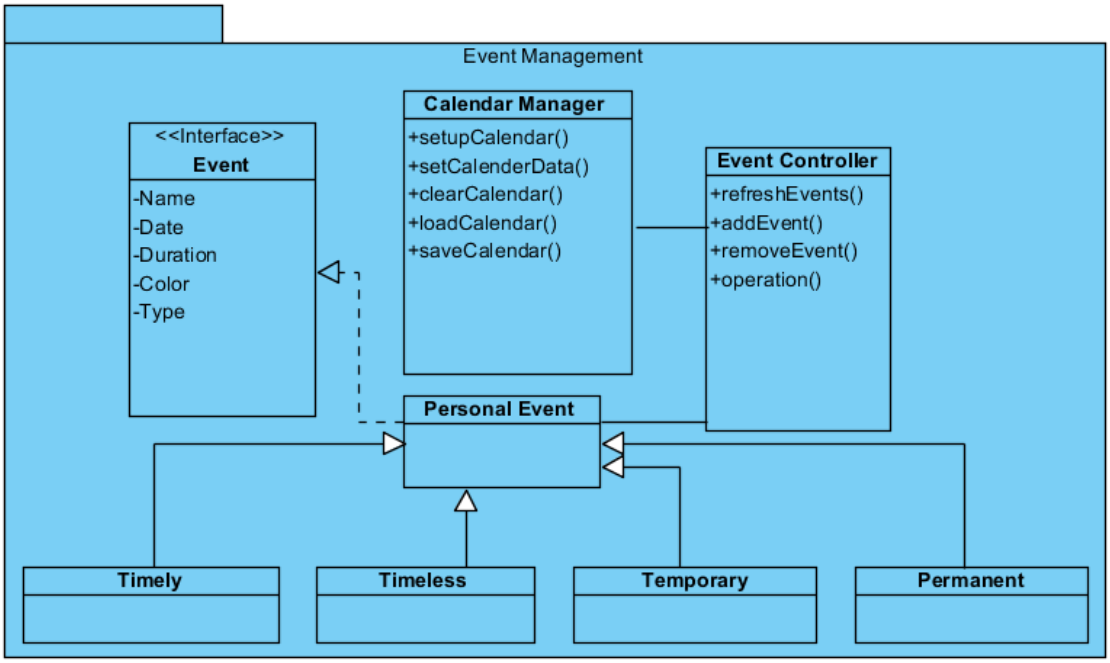

Figure 16 - Package Diagram of Event Management Subsystem

Event Management Subsystem is the controller subsystem of the calendar. It provides event interface to make event objects and EventConroller to controll the events.

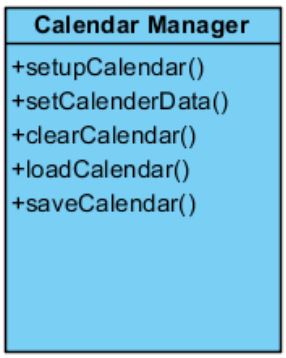

Figure 17 - CalendarManager Class

CalendarManager class the initializer class of the calendar. It uses data from the event objects and gets controller by the EventController.

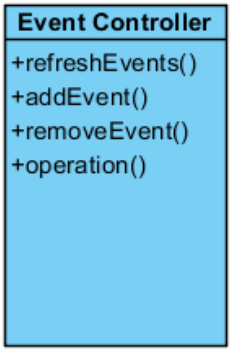

Figure 18 - EventController Class

EventController class controls the CalendarManager's activities by using other classes. This class can be used to refresh events, add new events or removeexisting events.

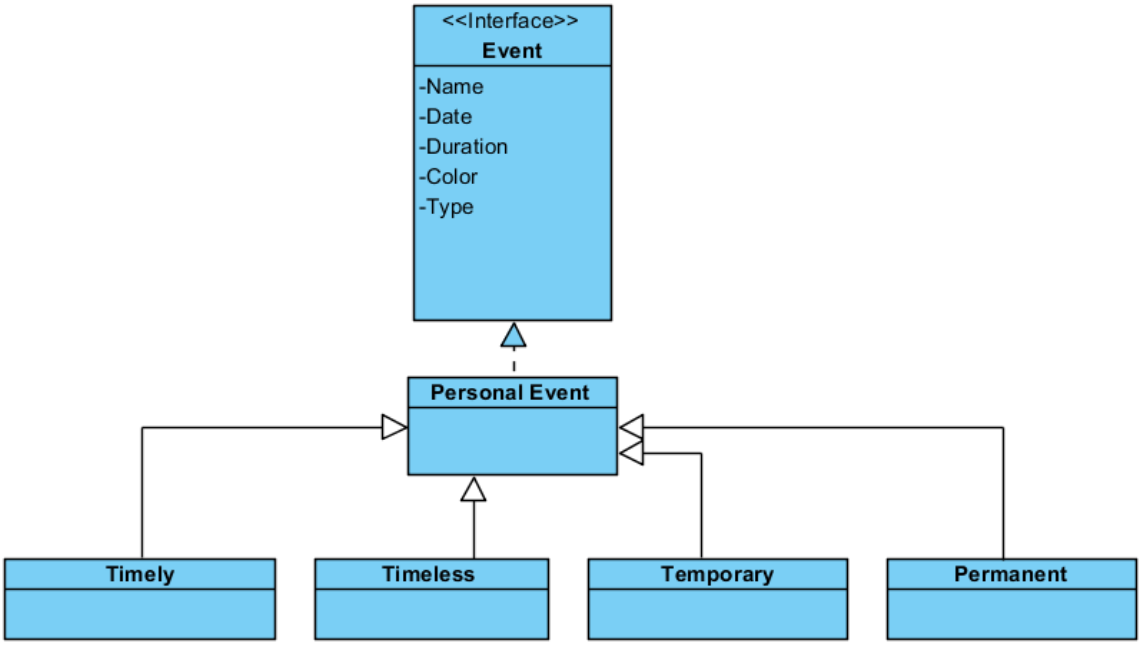

Figure 19 - Event Hierarchy in Event Management Subsystem

In the system, Event interface provides Personal Events and Smart Events as two different event types. Those event types have more features like being a timely event or being a timeless event, and at the same time being a temporary event or being a permanent event. However, the Smart Events will not be timeless events.

3.3.4 User Interface Subsystem Interface

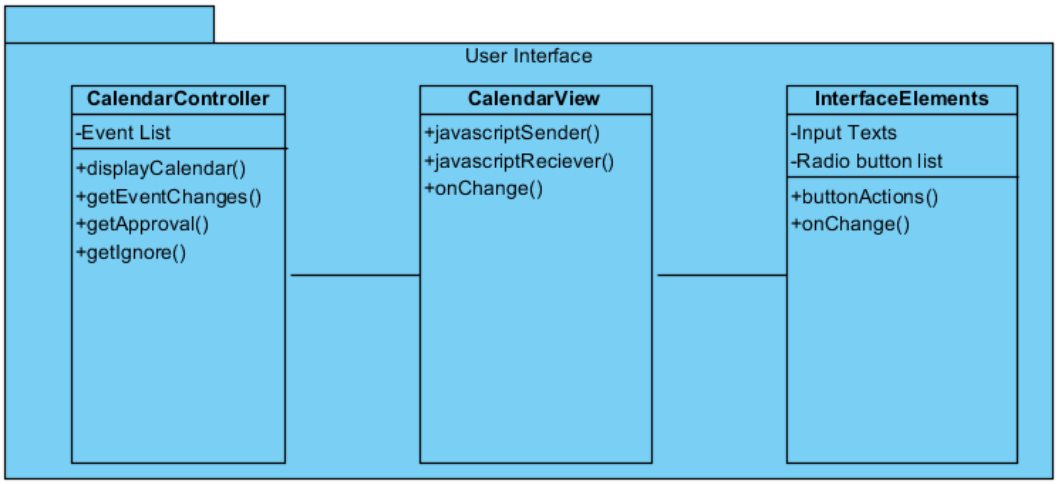

Figure 20 - Package Diagram of User Interface Subsystem

User Interface Subsystem displays the program on the users' devices. There are many interface elements to be added and once they are added they will communicate with the rest of the system.

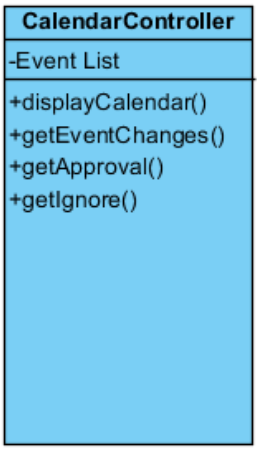

Figure 21 - CalendarController Class

CalendarController class have functions that are for getting event changes and approving or ignoring Smart Events. Plus, this class sends data of users' event lists to be displayed on the calendar

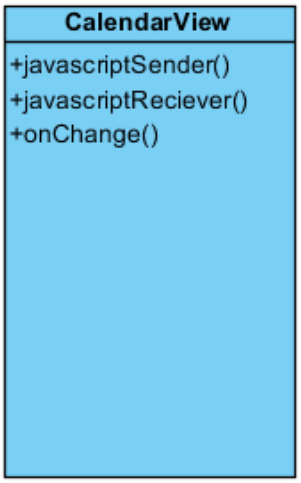

Figure 22 - CalendarView Class

CalendarView class uses data from the system to display the users' calendars on their displays.

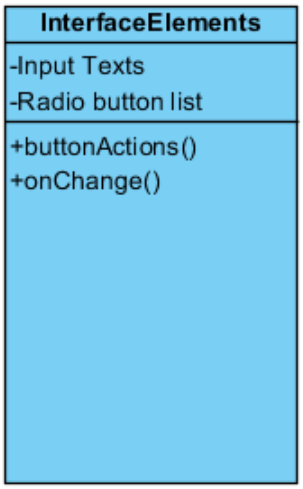

Figure 23 - InterfaceElements Class

This class actually have many child classes but they are far from finish. However, this class will use the child classes to show the other features like adding friend or logging out buttons on the users' displays.

3.3.5 Profile Management Subsystem Interface

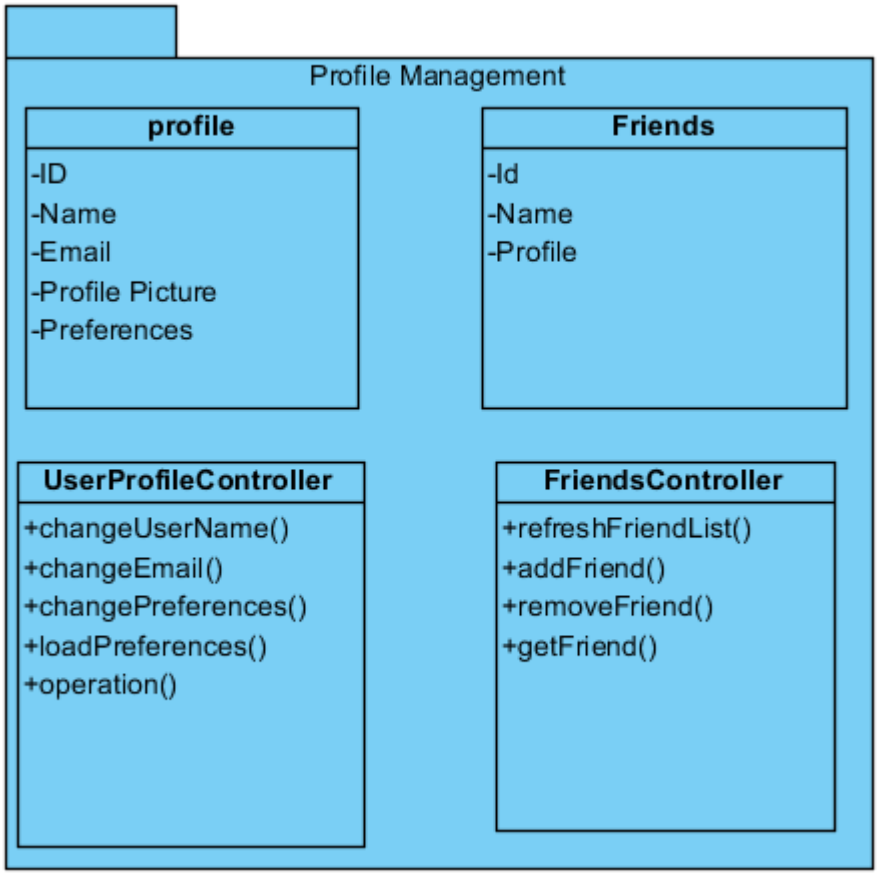

Figure 24 - Package Diagram of Profile Management

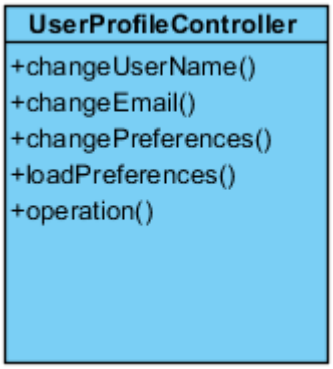

Figure 25 - UserProfileController Class

After registration, it is possible to one change recorded personal informations in the system. Username can be changed by changeUserName method, email address can be changed

by changeEmail method and preferences that contains choices such as “Who can see your profile?” or “Who can see your email number?” can be changed by changePreferences method.

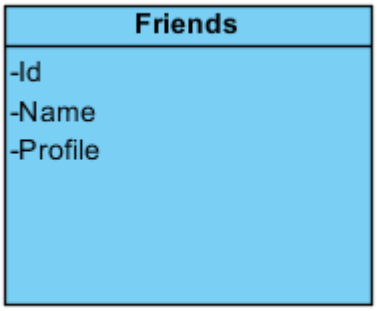

Figure 26 - Friends Class

Every friend has its own id, name and profile in the system and Friends Class holds these informations.

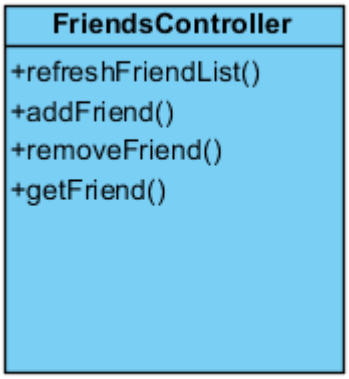

Figure 27 - FriendsController Class

It is possible to add new friends by addFriend method and remove existing person on friend list by removeFriend. To see changes in friend list, refreshing friend list is needed which is provided by refreshFriendList method.

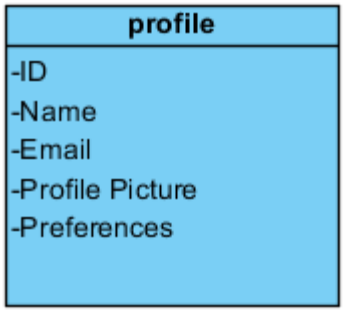

Figure 28 - profile Class

Like it is mentioned, a profile has its own id, name, email and prefernces. Also, every profile has a profile picture.

3.3.6 SmartEvent Management Subsystem Interface

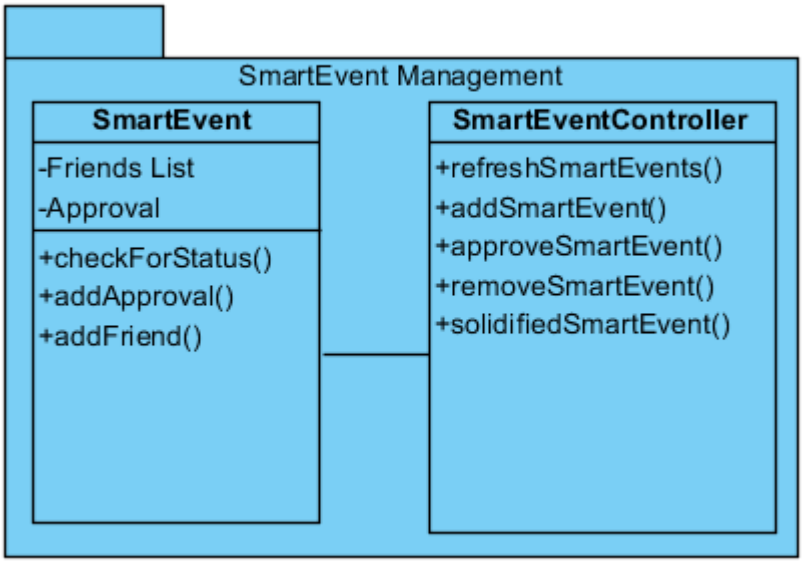

Figure 29 - Package Decomposition of SmartEvent Management Subsystem

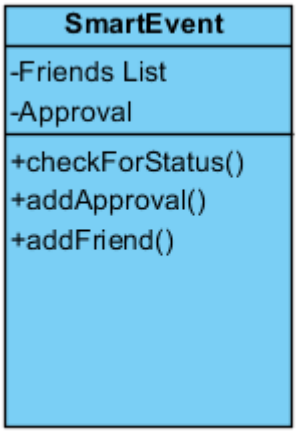

Figure 30 - SmartEvent Class

SmartEvent class holds one's friends list and a boolean approval which shows whether this SmartEvent is approved or not.

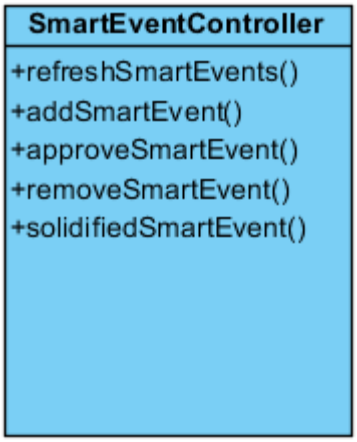

Figure 31 - SmartEventController Class

SmartEventController class is here to communicate with the calendar view. This class refreshes, adds, approves orremoves the Smart Events.

3.3.7 Dynamic Notification Management Subsystem Interface

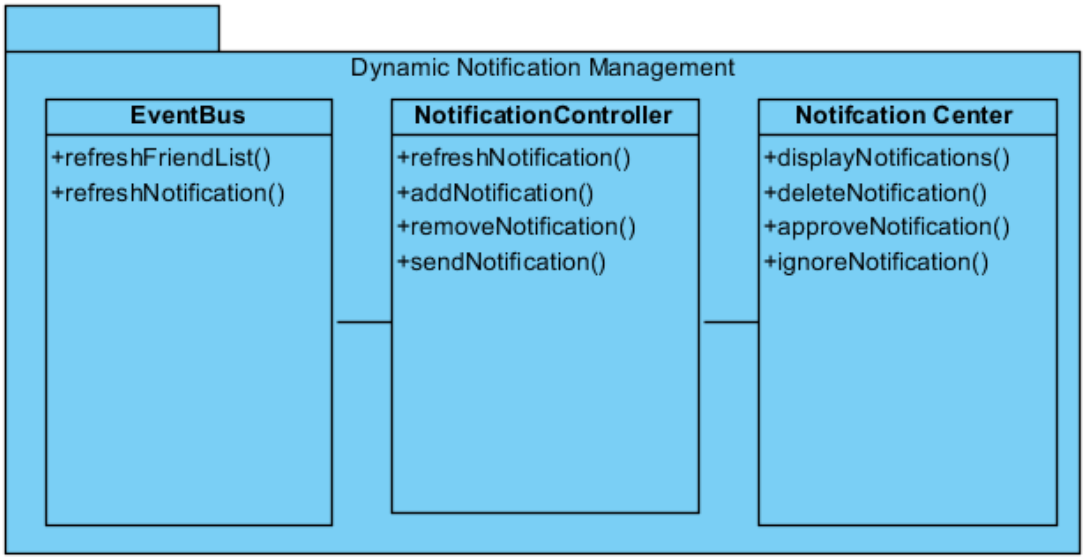

Figure 32 - Package Decomposition of Dynamic Notification Management Subsystem

Dynamic Notification Management Subsystem is active working subsystem. It get data from database and sends data to database whenever there is a internet connection available.

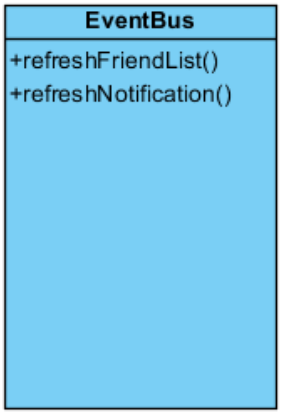

Figure 33 - EventBus Class

Eventbus class is a Publish & Subscribe Pattern used notification system. The class is using for triggering action which needed to be regularly triggered.

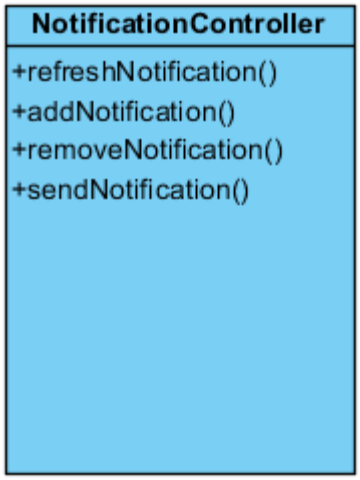

Figure 34 - NotificationController

NotificationController is used for adding and removing notification messages. Also, it sends messages to the database.

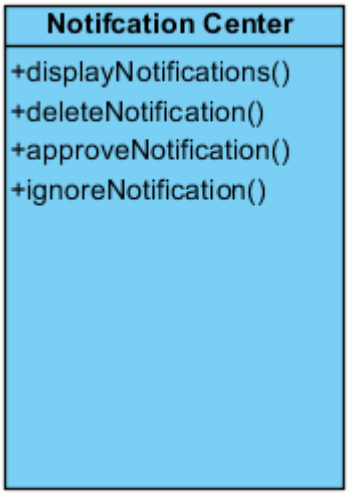

Figure 35 - NotificationCenter Class

NotificationCenter is a class for communication between user interface and notification subsystem. Also, it gathers user inputs as they are accepted or ignored.

4. References

[1] https://en.wikipedia.org/wiki/Java_virtual_machine

[2] https://techterms.com/definition/jre