Bilkent University

Department of Computer Engineering

Object-Oriented Software Engineering Project

CS 319 Project: Tempo

Analysis Report

Project Group: 1.G

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

Course Instructor: Eray Tüzün

Report

April 3, 2018

Table of Contents

Analysis Report

CS 319 Project : Tempo

1. Introduction

1.1 Purpose

This report is a description of a Software Requirements Specification (SRS) for our smart schedule application that is called “Tempo”. This report includes the features that we want to add to this application. These features are related to help people to schedule their times better. However, the main goal of his software These features will be mostly focused on one particular feature which is the thing that we call a “Smart Scheduling”. Furthermore, this report includes an overview of the application, describes the basics of the application by using UML: it describes functional requirements, non-functional requirements, use-case models including scenarios, object and class models, dynamic models, and mock-ups of our application’s user interface.

1.2 Content of the Project

Tempo is a smart scheduling app that combines to-do-list, scheduler and event calendar to help people organize their time with other people more efficient and more effectively. Tempo’s core idea is to enable people to arrange their meetings easily by seeing other individuals’ free times. Tempo also provides a help for self-improvement by reminding tasks that people should do, keeping track of people’s activities and statistics about them.

1.3 Terms and Definitions

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 Smart Event) or without them (that is called Personal Event). In addition, it has some other categories like timeless event, timed event, temporary event and permanent event; Tempo enables users to combine these categories for describing their event in the best way. For example, a Personal Event can be a timed and temporary event.

Events can be Smart Events or Personal Events:

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 requests for the Smart Event and then, other participants are notified for this meeting, its explanation, participants, time period. Other participants, who are invited to the event, 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' calendar. If any participant does not accept it, system informs 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. If the user says yes to this question, the Smart Event will still be added to other participants' calendars except the user that declined the Smart Event request.

Personal Event: This event is for the only user that makes it. 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.

Smart Events are always timed events but they can be temporary or permanent events. Personal Events can be timeless events (added to to-do list) or timed events, and at the same time timed events can be temporary or permanent events.

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.

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

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.

Combination of timeless event and permanent event is called "Habit" in our program. Habits are tracked by program and user can view information about the habit like streak, start date and some charts about it.

2. Overview

In general, Tempo is an application that uses social network idea for scheduling. Like other scheduling apps, this application will be also easy to understand and to use. Visuals will help users by making them check and perfect their time management. Users will be able to use this application by registering to the system. After registration process, people will be able to login and start benefiting the application's many features. Those features will be adding events to their calendar easily, adding their friends or coworkers for choosing the common free time and making an event that is suitable for everyone that wants or is forced to join it, and by doing these regular or irregular events users will be able to track their statistics and get an idea of what they should do to make their time management better by looking at those statistics.

2.1 Registration and Login

Users need to register or login to access their accounts. To register, every user needs a unique username and password. After that, he/she needs to give information about his / her name and surname, email address and type a password, and type the password again to eradicate mistakes about it.

2.2 Calendar

Calendar is main frame in Tempo since the user can see what user will do in this calendar with their start time and end time and when others are available. User can also change one activity time period by just dragging it to another time period. If this activity is a Smart Event, the system sends a message to other participants for this request and this activity's place does not change until all participants accept the request. However, if it is just a personal event, the user is free to change the time of this event.

2.3 Events

Events are core concept in Tempo. There are many types of events that are Smart Events, personal events, and timed events, timeless events, permanent events and temporary events. By combining these types of events, the user can describe an event in the best way and also user can add some stickers to them for visualization and quickness. For example, when a user looks at an event from the calendar and see that it has a sticker that shows an image of table, the user will understand that he has to be in a meeting at that time.

2.4 Smart Scheduler

Smart scheduler's main idea is adding some activities for a group. User can see free times of other possible participants so, user can select the best possible time period for their meetings. When time period is finalized, user (in this case it is called event admin) select other participants and in this way, they are notified about this meeting. When one participant accepts this meeting, that participant is added to meeting and system waits for other participants' response. When all participants accept it, it is added every participants' calendar. However, if one or more participant reject this request, system asks "Do you want to still arrange Smart Event without … ?" and event admin decides. If user says "yes", this event will be added to everyone that accepted this event's request. If user says "no", everyone that accepted this event will be notified about the cancellation of the event.

2.5 To - Do List

To - Do list is a list of timeless events and if user did this, he/she can click the checkbox to indicate whether he/she has done the event or not.

2.6 Friends List

Tempo enables users to see when their friends are available and when they are not. However, there is a huge possibility that one can have so many friends so, user can filter the ones who are not necessary to see their available and unavailable times and in this way, he is able to see the crucial ones really quick. This feature prevents a complicated appearance of calendar.

3. Requirement Specification

3.1 Functional Requirements

- Sign up to the system as a new user
- Authorization by login
- Display the calendar and events
- Create a Timed Event
- Create a Timeless Event
- Create a Permanent Event
- Create a Temporary Event
- Create a Habit Event
- View statistics of a Habit Event
- Mark completed for Timeless and Habit Events
- Remind the user by sending notification before the happening of a Event
- Relocate events by dragging through the calendar
- Delete events
- See masked schedule view of friends' calendar
- Request friends for a Smart Event
- Accept or ignore a Smart Event
- Notify if the Smart Event is approved or not
- Search friends with their username
- Chat with friends
- Update user profile
- Edit user preferences of the system

3.2 Non-functional Requirements

- Usability:
  - When a user send a Smart Event request, the user should see a masked schedules of his friends schedule with only time blocks and without seeing the details of their events to ensure privacy.
  - As soon as one participant in a smart event approves/cancels a request it need to be automatically updated in the event creator's profile. Hence there has to be background system running to ensure this.
  - The calendar need to have night mode option so that it is also very usable in night.
  - The calendar need to be usable by people as young as kindergarten kids (age ~8) to as old as 80 years old. Hence all of the features needs to be easily accessed and used. No screens can be too cluttered.
- Reliability:
  - The System will back up data from the local storage to the database every day so that a system crash does not result in data loss.
  - All the password and user information must be encrypted before stored in the database so that nobody can get access to sensitive information even if the database gets hacked.
- Performance:
  - The calendar app should not store more than 1MB data in the device and will to backup to the database as soon as the data reaches the limit. This will also clear the data stored in the device.
  - The system must be accessed at least by a million users without any system crashes, and database problems.
  - The system's some features such as looking at calendar or adding personal or timeless events must be usable even if the user does not have a internet connection. However, the user should be already be logged in before the offline use.

    - The total file size of the application needs to be less than 10MB so that it is fast to download and doesn’t take up much device memory.
    - The starting loading time of the app has to be less than 3 seconds.
- Maintainability:
    - The implementation has to follow the MVC model so that it would be easier in the later stage to integrate a lot choices for calendar view without changing the model and controller classes.
- Open-source:
    - The whole project need to be uploaded to Github so encourage collaboration and promote open-source environment before the deadline.

4. System Models

4.1 Use-Case Model

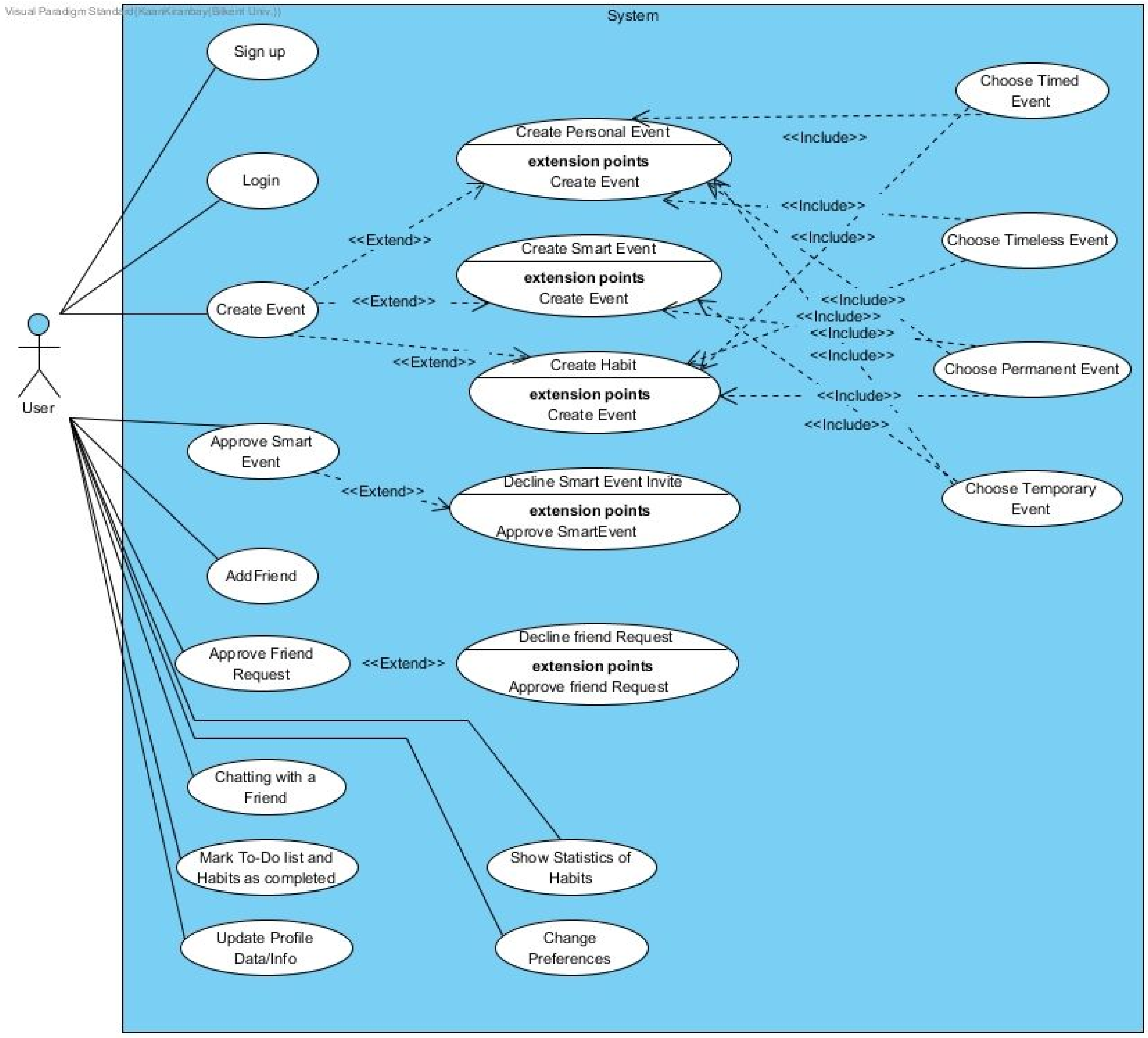


Figure 1 - Use Case Diagram

Use case name: ***Sign Up***

Participating actors: User

Flow of events: 1. The user enters his name, surname, email and password.
2. The system stores this information in the database and logs user in.

Entry condition: First time use of a new user

Exit condition: The user has provided the database with a unique username and an appropriate password.

Quality requirements: The password must be at least 6 characters long. And the password is case-sensitive. The username must be unique.

Use case name: ***Login***

Participating actors: User

Flow of events: 1. The user enters his username and password.
2. The system checks this information against database and logs user in.

Entry condition: Must be signed up.

Exit condition: The user's username and password has matched with of the UserInfo of the database.
Alternative Flow of events: If the username or password doesn't match, a warning is given and the user is informed to try again.

Use case name: ***CreatePersonalEvent***

Participating actors: User

Flow of events: 1. The user clicks “create event” and specifies the time and date.
2. The system creates the event and displays on the calendar.

Entry condition: The user is logged in.

Exit condition: The user has created a new event which is displayed on his calendar.

Alternative Flow of events: If the user clicks the checkbox repeat, this event is repeated accordingly. If not, the event will be added as a timed event to the calendar. If the user clicks on timeless event, then the event is added as a timeless event section(to-do list) instead of calendar.

Use case name: ***CreateSmartEvent***

Participating actors: User

Flow of events: 1. The user selects a group of friends.
2. The system responds by showing a superimposed version of the friends’ calendars on his calendar and free time available.
3. The user chooses a time from available spot.
4. The system sends all of the selects friends approval request.

Entry condition: The user is logged in.

Exit condition: The user has sent his selected friends a request for a timed event.

Use case name: ***ApproveSmartEvent***

Participating actors: User

Flow of events: 1. The system display a requested event from a friend on your calendar.
2. You click approve.
3. The system sends this information to that friend's Smart Event object.

Entry condition: CreateSmartEvent request from a friend.

Exit condition: The user responded by expressing his approval or disapproval.

Use case name: ***AddFriend***

Participating actors: User

Flow of events: 1. The user searches the other user and clicks his username.
2. The system shows the profile of the other user.
3. The user clicks the add friend button.
4. The system responds by sending friend request to the other user.

Entry condition: The user is logged in.

Exit condition: The user requested and if approved the users are added to each other's friend list, and the user that added another user as a friend will be notified.

Alternative Flow of events: If the request is not accepted, no friend is added, and the user that wanted to add another user as a friend will be notified.

Use case name: ***ApproveFriendRequest***

Participating actors: User

Flow of events: 1. The system display a friend request from a user.

2. You click approve.

3. The system sends this information to that friend and saves them as friends.

Entry condition: AddFriend request from a friend.

Exit condition: The user responded by expressing his approval or disapproval.

Alternative Flow of events: If the other user doesn't accept the request, then no friend is added.

Use case name: ***Chatting with Friends***

Participating actors: Users

Flow of events: 1. The user selects a friend from the friend list and clicks on it.

2. The system opens a new pop-up window for user that shows chat with the selected friend.

3. The user types something to the chat windows' writing section, and sends it to his friend.

4. The system stores what the user written into the database and shows it in the user's friend's chat.

5. The user's friend writes anything back to user.

6.The system stores what the user's friend written into the database and shows it in the user's chat as well.

This goes on until the chat is closed. However, when the chat is closed the previous conservations are still stored in the database.

Entry condition: User must have a friend added.

Exit condition: User closes the pop-up window.

Alternative Flow of events: If the any of the users in the chat loses internet connection, messages will not be sent anytime.

Use case name: ***Mark To-Do list and Habits as completed***

Participating actors: User

Flow of events: 1. The system displays to-do/habit list.
2. User clicks on a box near a item inside of the to-do/habit list to make the to-do/ habit as completed.
3. The system stores the to-do/habit as completed and adds it to the statistics of the user.

Entry condition: The user should have at least an event in to-do/habit list.

Exit condition: The system adds the statistics to the database then exit.

Alternative Flow of events: If the user doesn't mark it as completed, the system still stores the statistics but not as completed but uncompleted which again affects the statistics of the user.

Use case name: ***Show Statistics of Habits***

Participating actors: User

Flow of events: 1. The user clicks on "Statistics" tab via profile window.
2. The system shows the statistics and graphs of each habit in the new tab.

Entry condition: The user should be logged-in to the system.

Exit condition: The user clicks on another tab or closes the profile window.

Use case name: ***Change Preferences***

Participating actors: User

Flow of events: 1. The user clicks on “Preferences” tab via profile window.

2. The system displays a tab that shows the preference options of the user.

3.The user changes the preferences as he/she likes then clicks “Save”.

4. The system saves the user’s preferences to the program on user’s PC.

Entry condition: The user should be logged in to the system.

Exit condition: The user clicks on another tab or closes the profile window.

Use case name: ***Update Profile Data/Info***

Participating actors: User

Flow of events: 1. The user clicks on “Profile” window.

2. The system shows the user profile data and what can be changed by the user.

3. The user changes his/her profile data or uploads a new profile image then clicks “Save”.

4. The system saves the user’s preferences to the program on the database.

Entry condition: The user should be logged-in to the system.

Exit condition: The user clicks on another tab or closes the profile window.

4.2 Object and Class Model

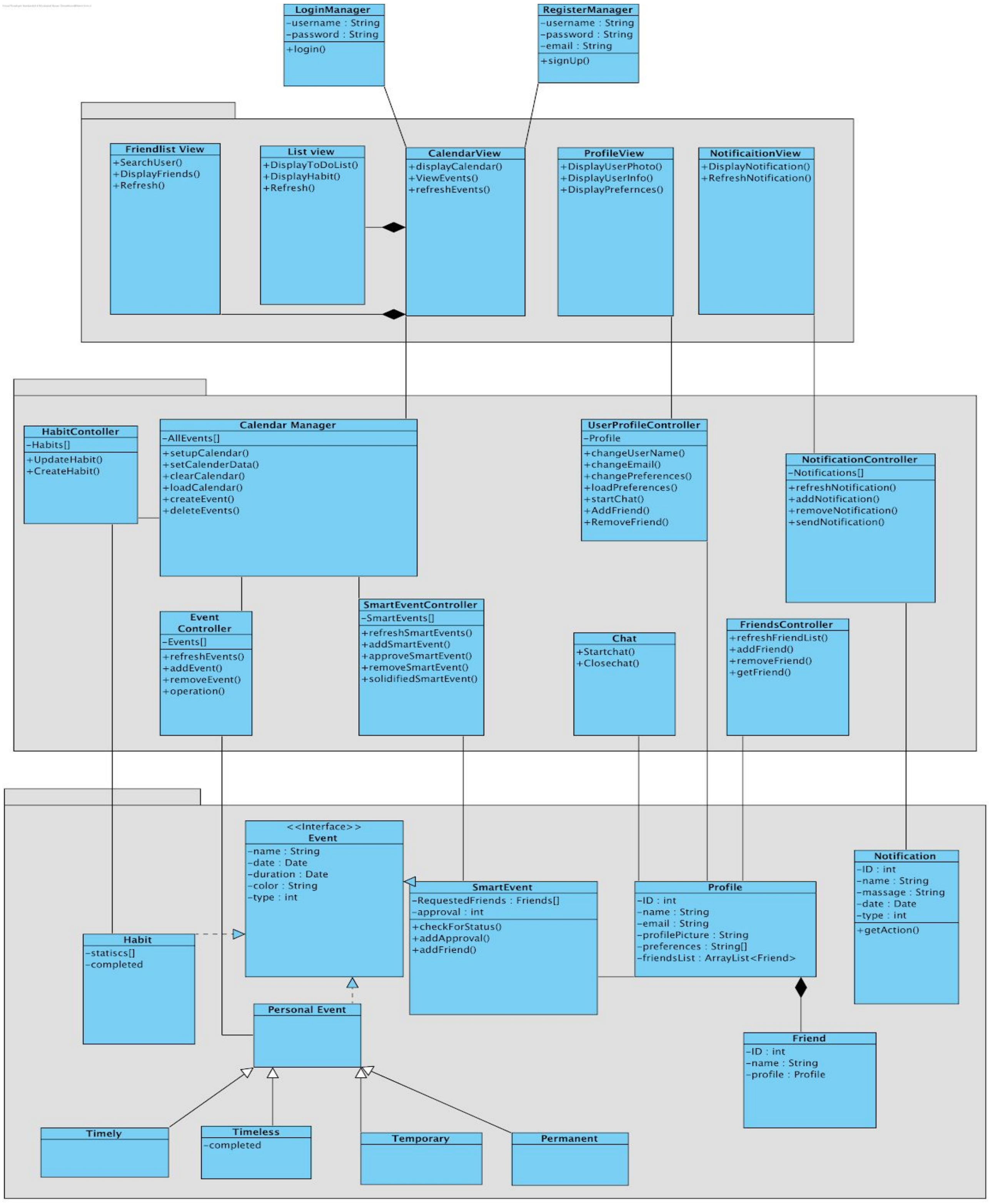


Figure 2 - Class Diagram

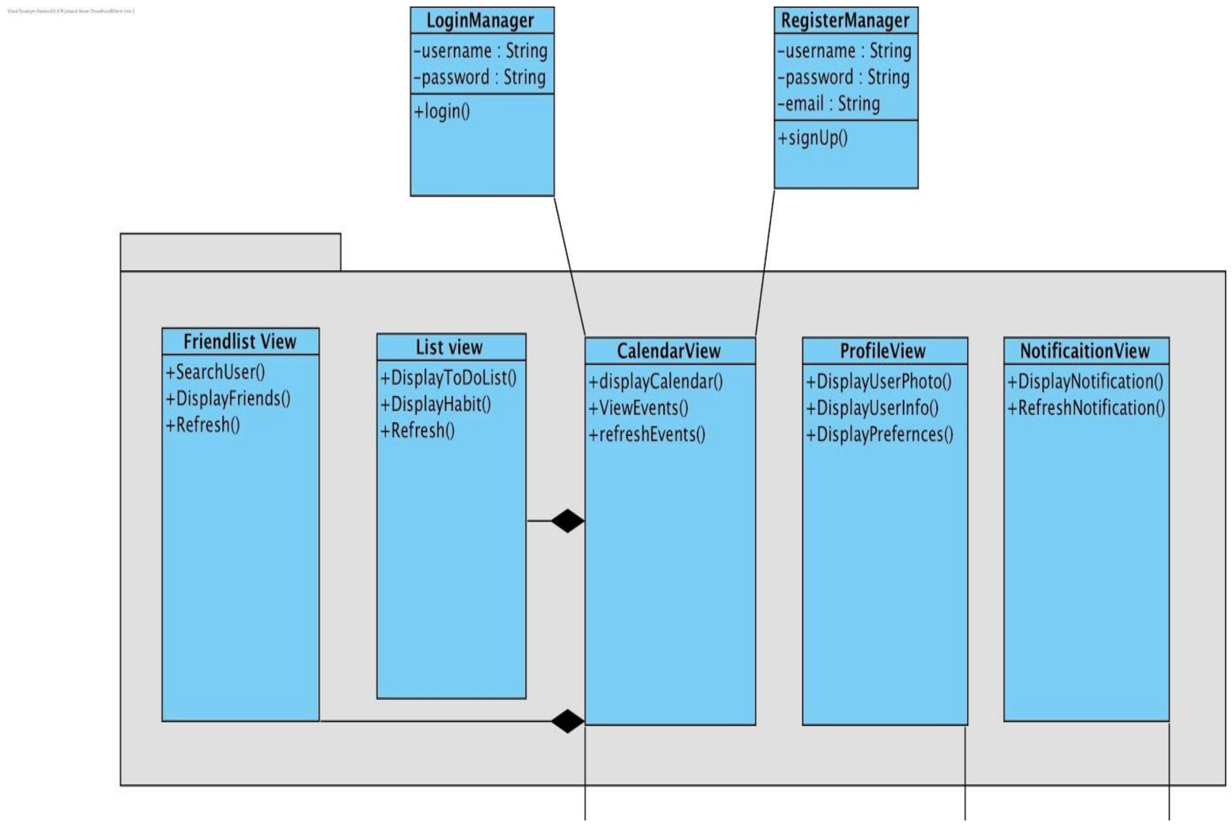


Figure 3 - View Package in Class Diagram

Views contains all the views that the system requires. It is the top level in the MVC hierarchy. The views only interact with the controllers and displays the contents. The user interacts with the system using only the views. The calderview displays the calendar and all the events except timeless and habit events. List View displays timeless and habit events. Friendlist View Displays the friends list. Profile View displays the user profile. Notification view display notifications.

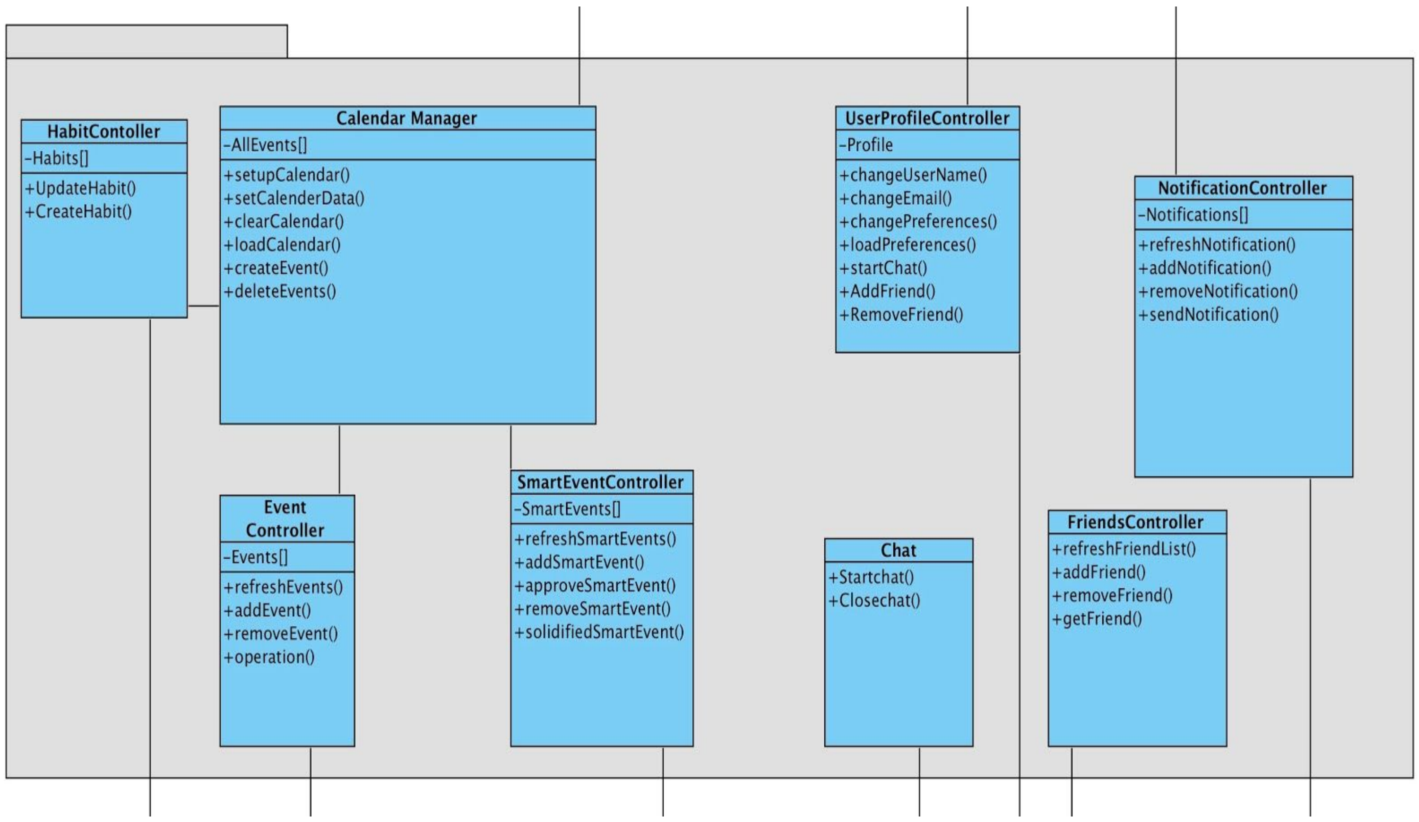


Figure 4 - Controller Package in Class Diagram

Controller classes modifies the data objects and sends them to the views for display. Calendar Manager creates events using event controllers and sends to the calendar view for display. Habit controller, Event Controller, SmartEvent Controller controls various types of events. UserProfileController class controls the profile object and sends it to ProfileView. Chat is also initialized by UserProfileController. NotificationController controls notificaiton object and sends them to NotificaitonView.

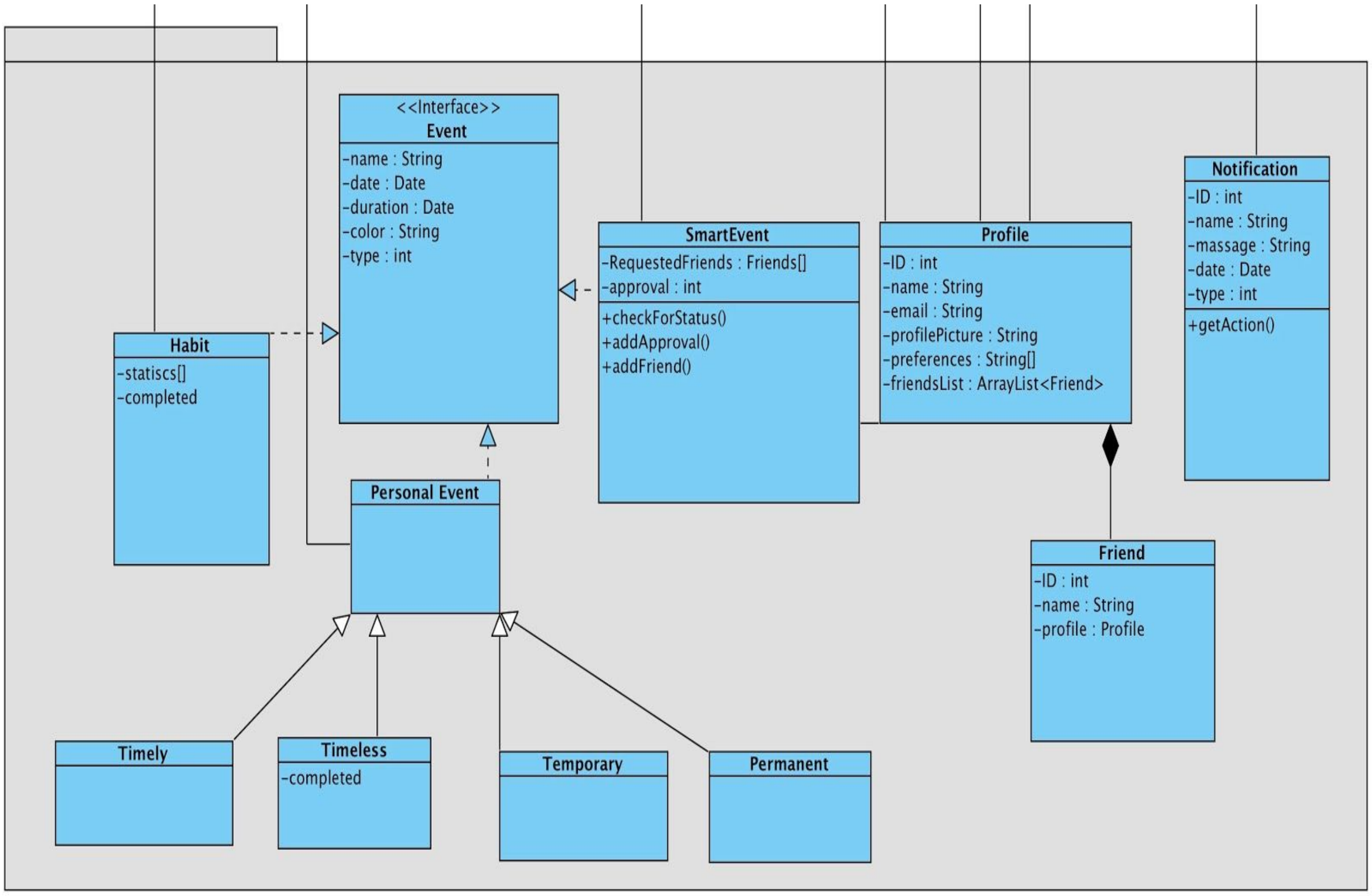


Figure 5 - Model Package in Class Diagram

Model classes stores all the data related to the system. Various types events are stored by the habit, timely, timeless, temporary, permanent, smartevent classes. Profile info, friendlist and preferences are stored by the profile class. Notification stores data related to a single notification.

4.3 Dynamic Models

This part of the Analysis Report contains some crucial and detailed information about the Tempo by using dynamic models such as sequence diagrams and activity diagrams.

4.3.1 Sequence Diagrams

This part focuses on how starting the application occurs, how adding events to calendar occurs, how adding a common event to calendar occurs, and how adding a friend occurs by using sequence diagrams.

4.3.1.1 Add Smart Event

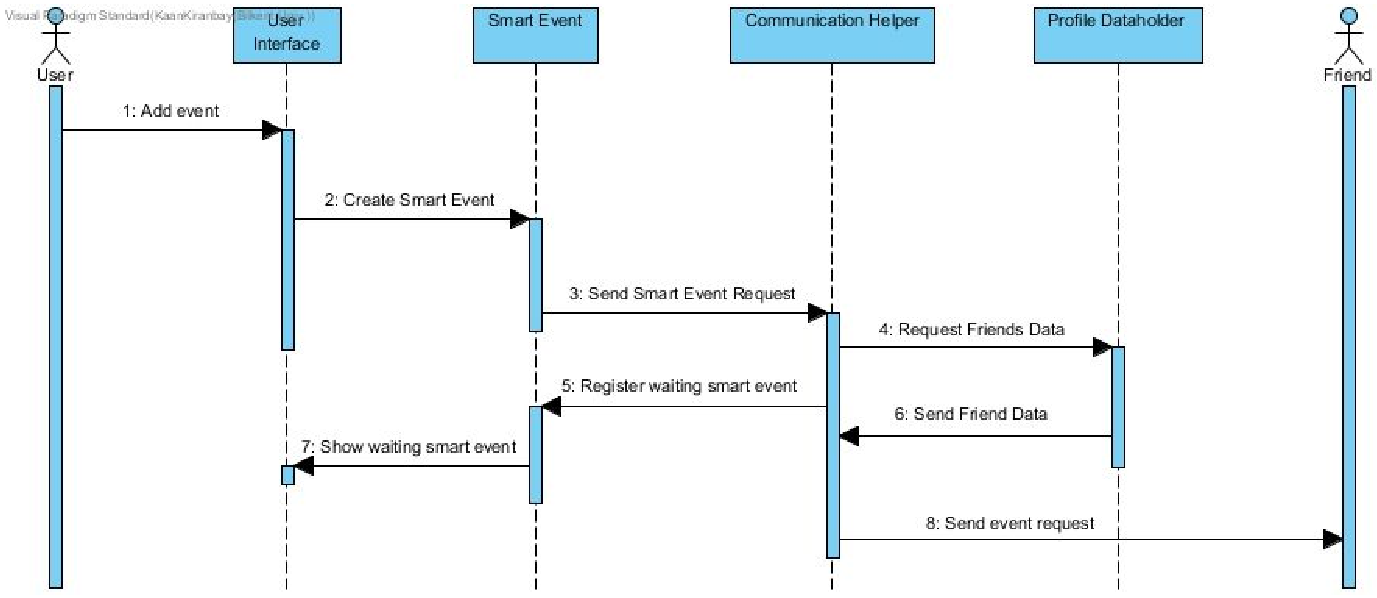


Figure 6 - Adding a Smart Event to the Calendar Sequence Diagram

Above sequence diagram (figure 4) illustrates the scenario explained below:

<u>**Scenario:**</u> User clicks “Add Event” button once. A new window will open and user choose type of an event as a SmartEvent. After that, participants are indicated by clicking on them and request is sent.

4.3.1.2 Approve Smart Event Request

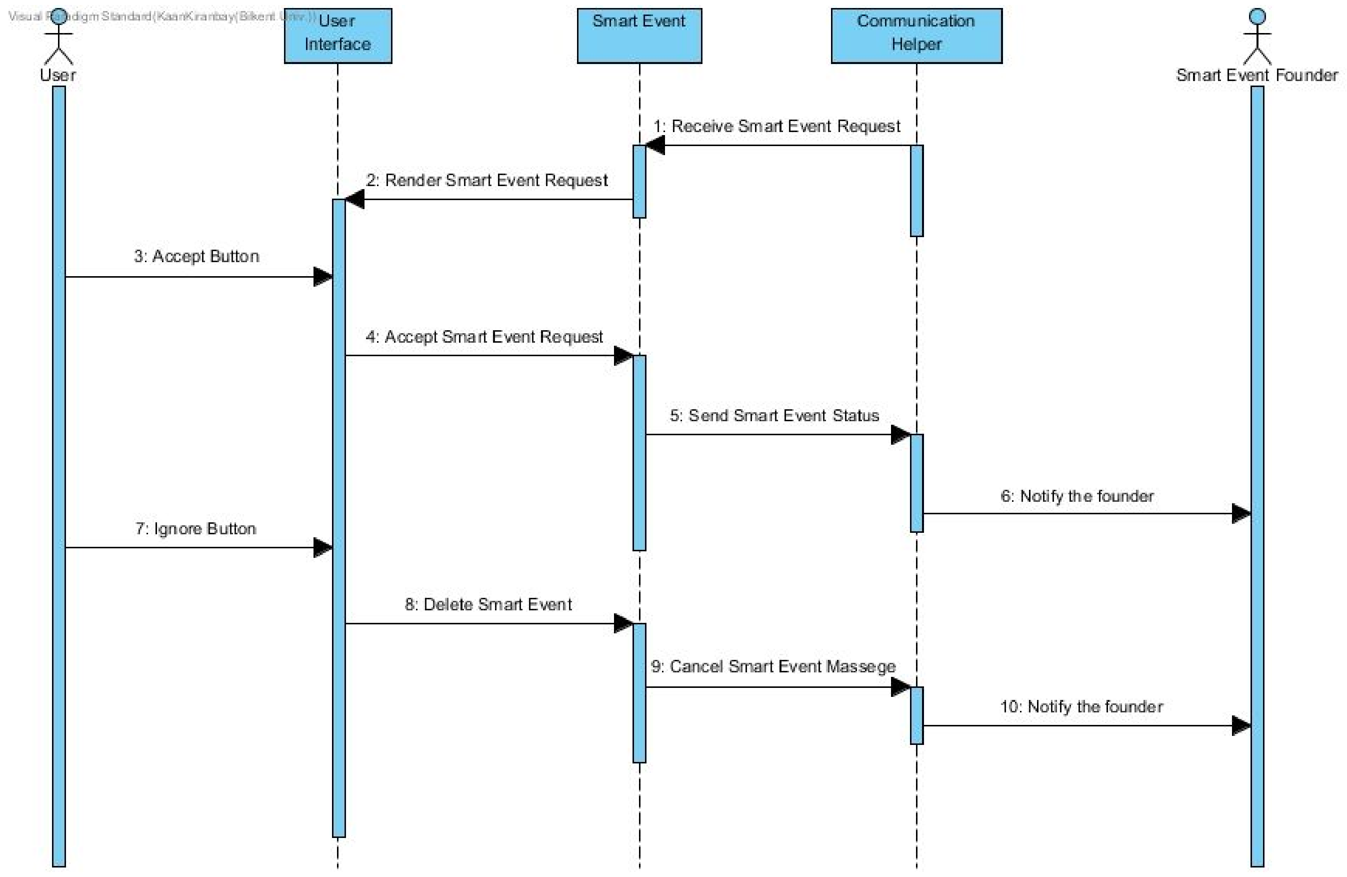

Visual Paradigm Standard(KaanKiranbay(Bilkent Univ.))
User
User Interface
Smart Event
Communication Helper
Smart Event Founder
1: Receive Smart Event Request
2: Render Smart Event Request
3: Accept Button
4: Accept Smart Event Request
5: Send Smart Event Status
6: Notify the founder
7: Ignore Button
8: Delete Smart Event
9: Cancel Smart Event Massege
10: Notify the founder

Figure 7 - Accepting a Smart Event to the Calendar Sequence Diagram

Above sequence diagram (figure 5) illustrates the scenario explained below:

Scenario: When user receive a Smart Event request, a notification will be displayed and user can either accept or ignore it. For both of the response, Smart Event Founder will be notified according to user's response.

4.3.1.3 Add Friend

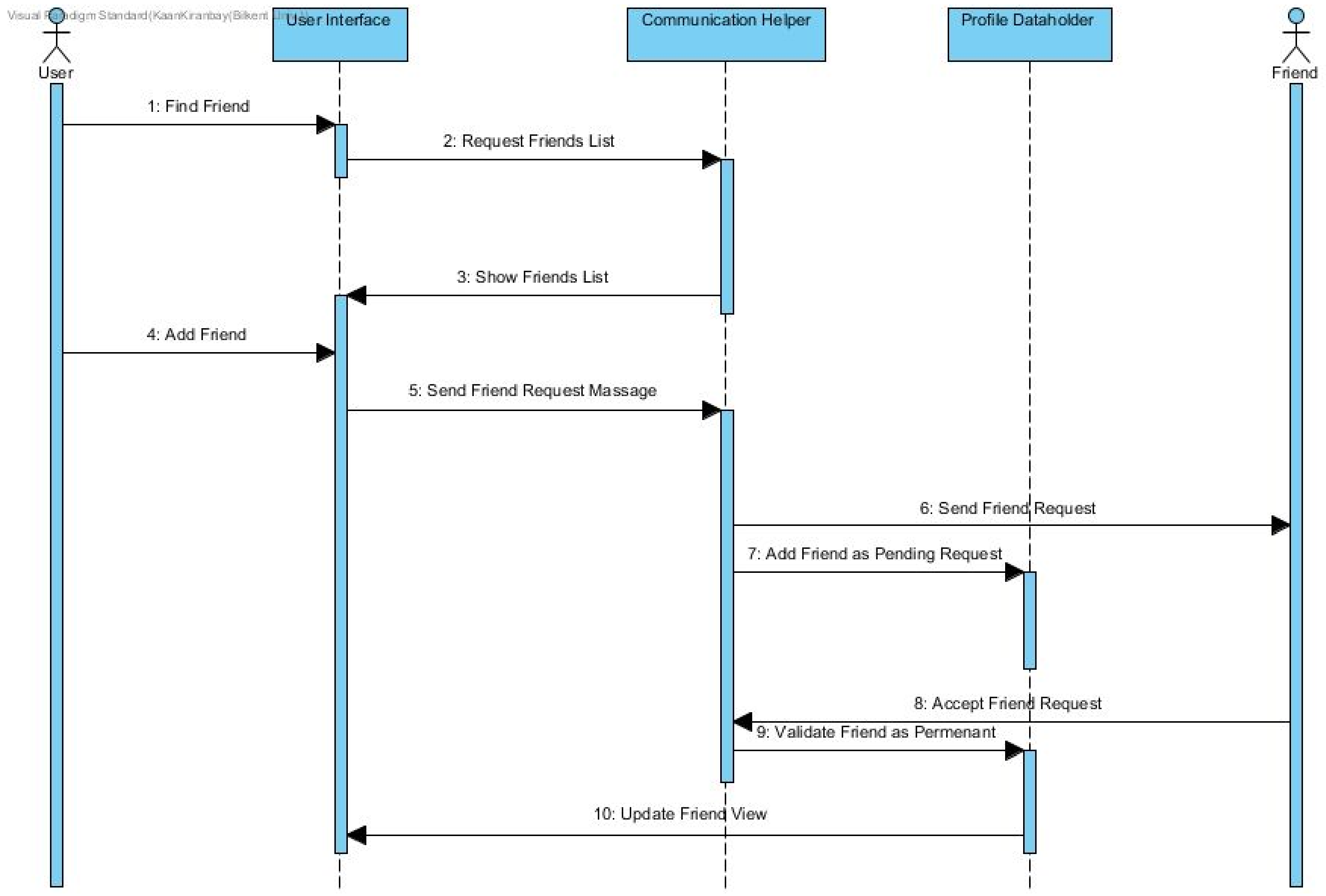


Figure 8 - Adding a Friend Sequence Diagram

Above sequence diagram (figure 6) illustrates the scenario explained below:

Scenario: In menu (Statistics/Settings) page, user can access "Add Friend" button. After clicking it, first he needs to find the friend by search bar. After that, by clicking "Add Selected Button", friend request send to the corresponding friend. If the friend accepts the request, user can see the friend in his friend list and also see his activities on calendar.

4.3.1.4 Approve Friend Request

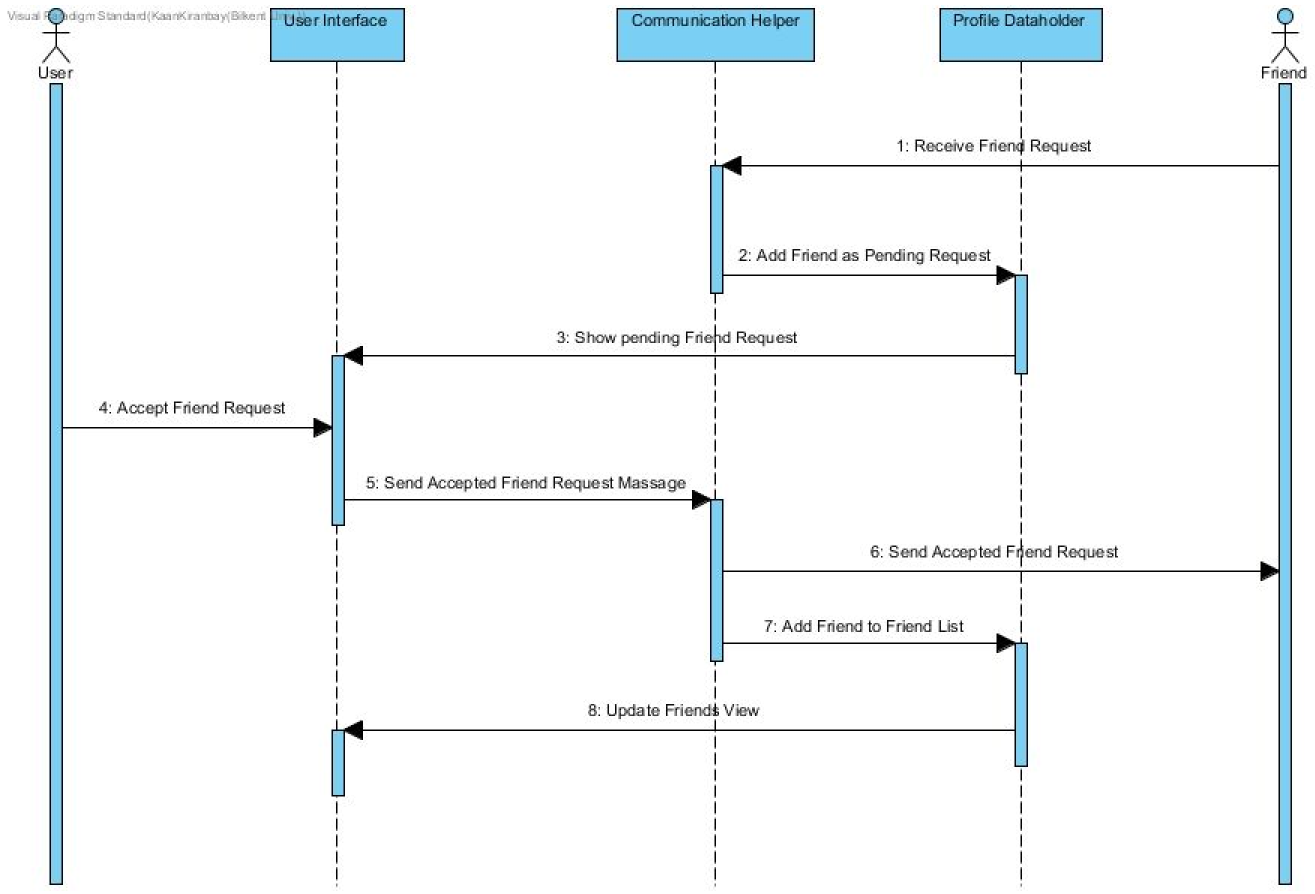

Visual Paradigm Standard(KaanKiranbay(Bilkent Univ.))
User
User Interface
Communication Helper
Profile Dataholder
Friend
1: Receive Friend Request
2: Add Friend as Pending Request
3: Show pending Friend Request
4: Accept Friend Request
5: Send Accepted Friend Request Massage
6: Send Accepted Friend Request
7: Add Friend to Friend List
8: Update Friends View

Figure 9 - Approving Friend Request Sequence Diagram

Above sequence diagram (figure 7) illustrates the scenario explained below:

Scenario: After having a friend request from a friend, user will be notified about it. If user accepts the request, friend will be displayed on his friend list and also his activities can bee seen on the calendar. After user accepts the request, friend also will be notified about this response.

4.3.1.5 Show Habit Statistics

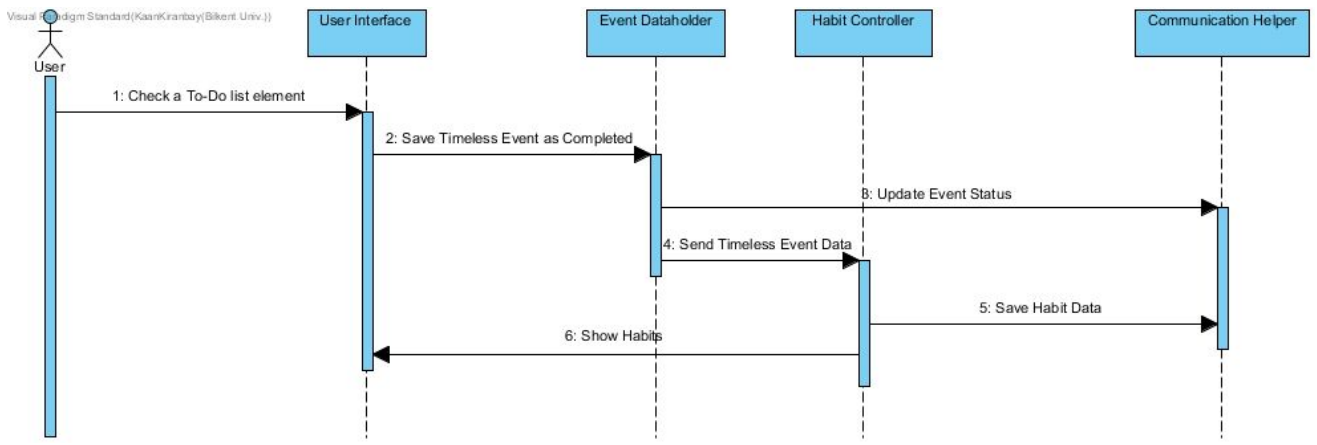

Visual Paradigm Standard(KaanKiranbay(Bilkent Univ.))
User
User Interface
Event Dataholder
Habit Controller
Communication Helper
1: Check a To-Do list element
2: Save Timeless Event as Completed
3: Update Event Status
4: Send Timeless Event Data
5: Save Habit Data
6: Show Habits

Figure 10 - Showing Habit Statistics Sequence Diagram

Above sequence diagram (figure 8) illustrates the scenario explained below:

<u>Scenario:</u> User can access habit statistics in the statistics tab. In this tab, user can see recent records and statistics of his corresponding habit which can be selected.

4.3.2 Activity Diagram

This part focuses on activities of the system during the usage of application are also shown in the activity diagram.

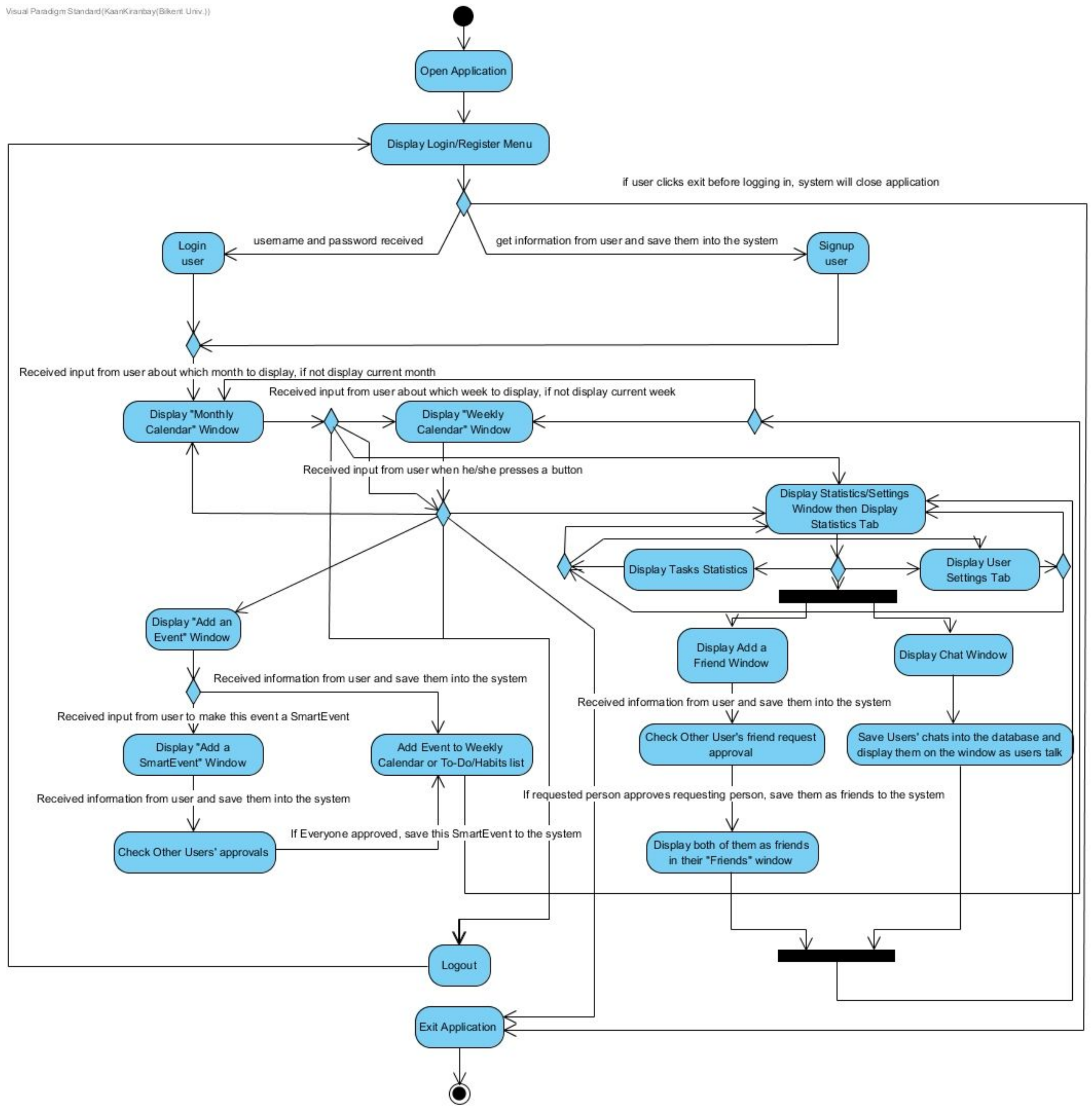

Visual Paradigm Standard(KaanKiranbay(Bilkent Univ.))
Open Application
Display Login/Register Menu
if user clicks exit before logging in, system will close application
Login user
username and password received
get information from user and save them into the system
Signup user
Received input from user about which month to display, if not display current month
Received input from user about which week to display, if not display current week
Display "Monthly Calendar" Window
Display "Weekly Calendar" Window
Received input from user when he/she presses a button
Display Statistics/Settings Window then Display Statistics Tab
Display Tasks Statistics
Display User Settings Tab
Display "Add an Event" Window
Display Add a Friend Window
Display Chat Window
Received information from user and save them into the system
Received input from user to make this event a SmartEvent
Display "Add a SmartEvent" Window
Add Event to Weekly Calendar or To-Do/Habits list
Received information from user and save them into the system
Check Other User's friend request approval
Save Users' chats into the database and display them on the window as users talk
Received information from user and save them into the system
If requested person approves requesting person, save them as friends to the system
Check Other Users' approvals
If Everyone approved, save this SmartEvent to the system
Display both of them as friends in their "Friends" window
Logout
Exit Application

Figure 11 - Activity Diagram

This activity diagram (figure 7) indicates how the system works and flows. It illustrates the description of the application explained below:

A user will open the application by clicking on the application’s icon on the desktop. System will initialize and display the main menu. Then the user will have 3 options: to login to the system or to sign up to the system or exit the application. When user logs in to the system, user will be able to use the application and its features. Logging the user in to the system will be by getting username and password from user and then system will check if the person exists or doesn’t. If the person exists, then system will show them their choice of calendar. If not, then system will ask them to register. Registering the user in to the system will be by getting their name and surname, username, password, and e-mail. After registering and saving the information of the user to the system, system will open the application automatically for them. Then the usage of application will begin.

After logging in, the system will display monthly calendar. The system will automatically open the current month as default. The user will have 5 options: Logging out, displaying the weekly calendar, adding or deleting an event, and displaying the statistics/habits window.

The user can be able to use a button to open weekly calendar, the system will also automatically open the current week as default. However, after pressing the button to logout, the system will display the main menu window again.

In both calendar window, the user will be able to add events to calendar or to-do/habits list by clicking a add event button. After clicking the button a new page will pop-up window will open and the user will be able to choose if the event is going to be a personal event, a smart event or a habit. After user chooses what the specifications of the event is it will be displayed in the calendar window.

In calendar window, the user can go to statistics/settings menu by clicking a button. After clicking it the calendar window will give its place to statistics/settings window. In here, user will be able to look his statistics about his task and habits, will be able to change program settings, will be able to add/remove friends, and will be able to chat with friends via friend list..

If the user clicks on add/remove friends, a new pop-up window will open to search people. After the user finds his friend, the user will be able to sent a request by clicking a button to add friend. If the user that is getting the request approves or declines the request, the user will be notified.

If the user clicks on a friend on his friend list, a new pop-up window will open. The user will be able to write whatever he likes in this window and sent it to his friend. These chat data will be saved to the database by the system.

After logging out from the system, users will be able to exit the application. The system will close the application if the user exits the app.

5. User Interface

5.1 Navigational Path

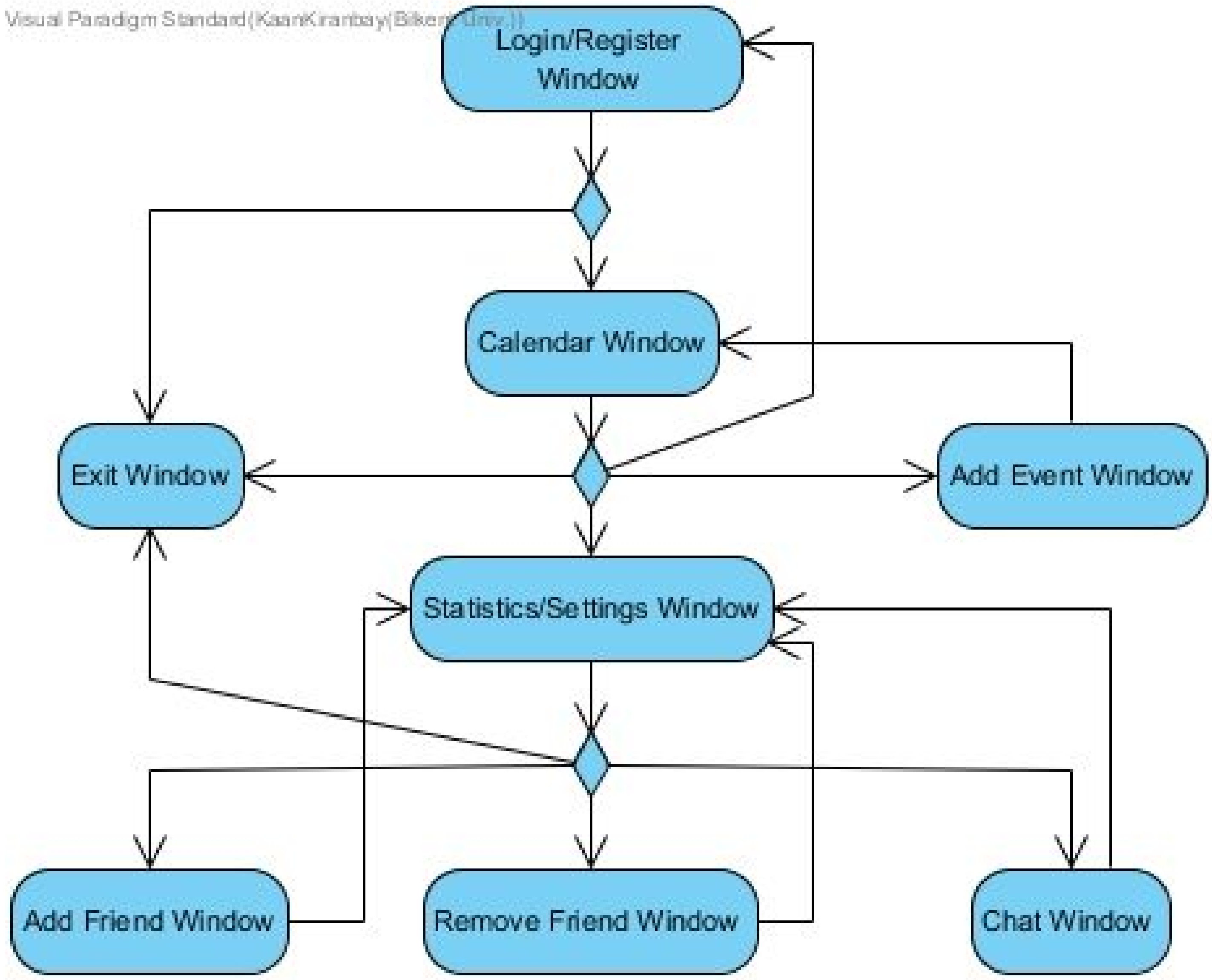


Figure 12 - Navigational Path

5.2 Screen Mock-ups

5.2.1 Opening Page

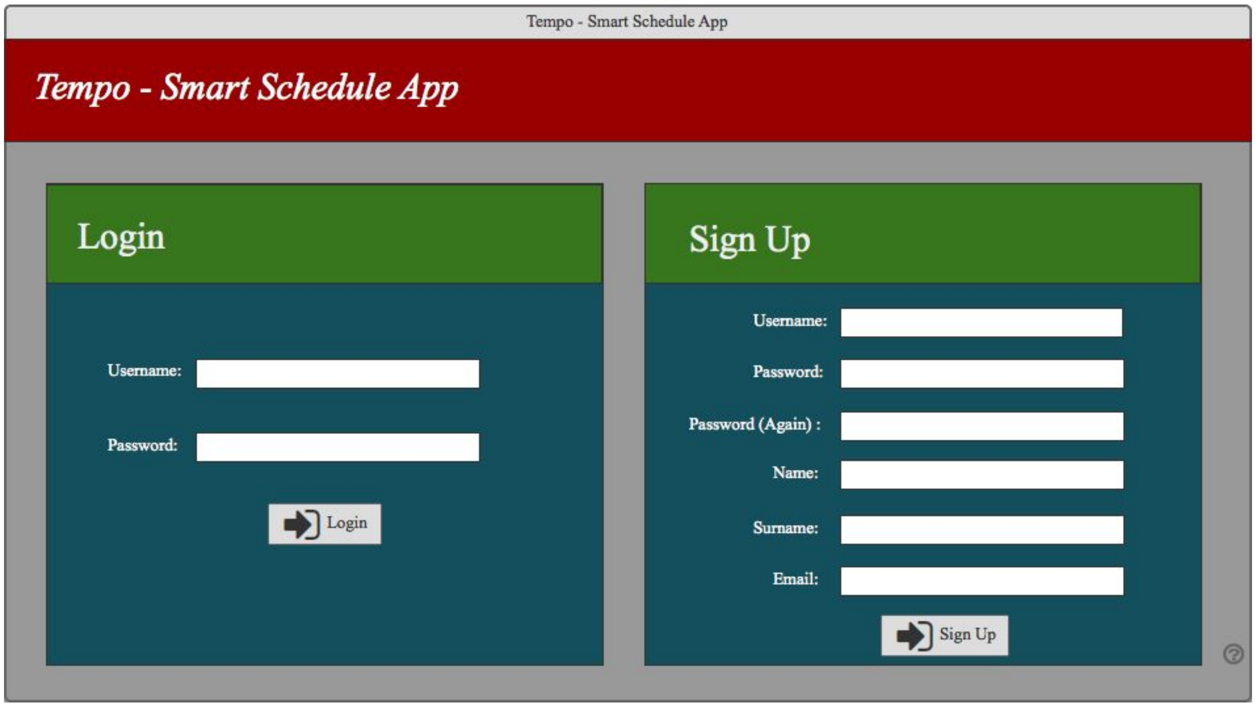


Figure 13 - Login/Register Window Mock-up

When user opens the app, user encounter opening page that contains Login and Sign Up pages. If user has no account, registration is needed. User need to choose a unique username and password. Then, for elimination of mistakes, user need to type password again. System checks whether they are equal. After that user needs to give information about name, surname and email. Final step is clicking to sign up button and user is signed up!

On the other hand, if user already has an account, typing username and password is needed to access profile.

5.2.2 Month Appearence of Calendar

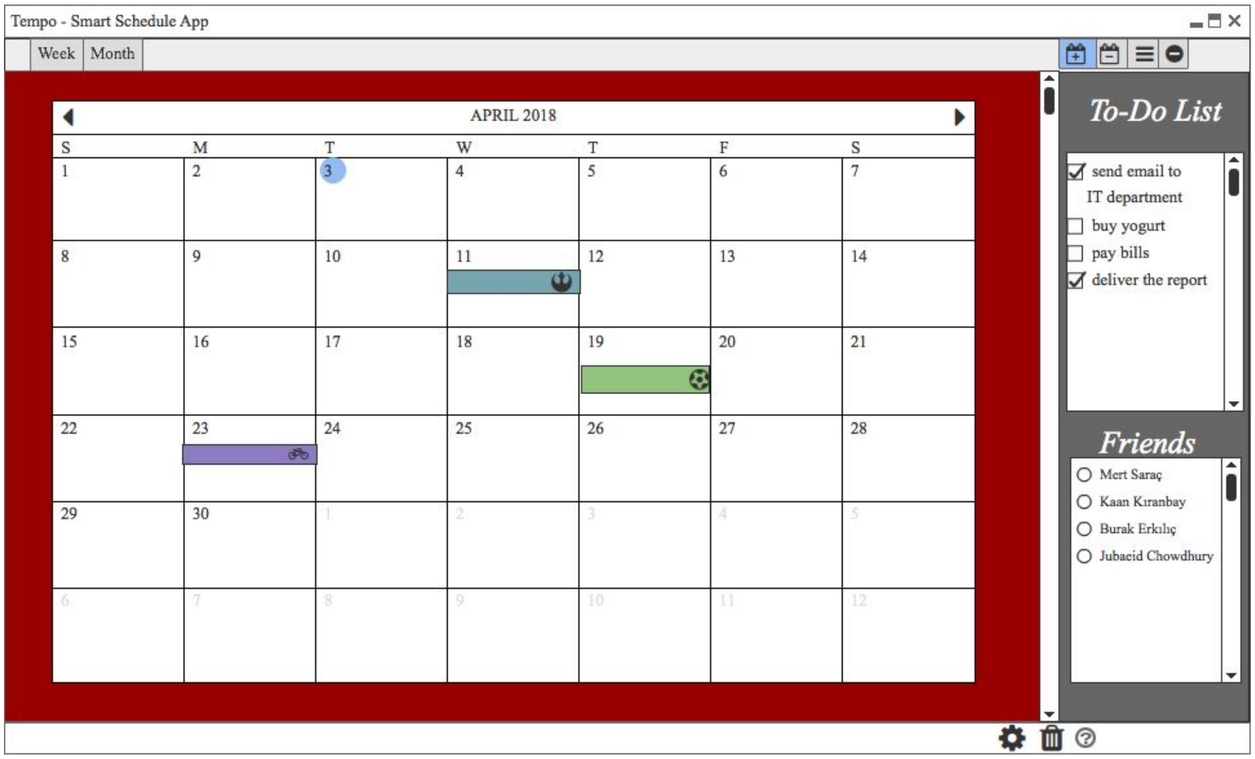


Figure 14 - Calendar Window Month Tab Mock-up

Opening page of Tempo is calendar page that is also main menu. From this menu, user can see all activities in general but not detailed for all 4 week of the month. If user wants to more detailed view for schedule, there is a week button to see detailed view of that current week. There are also 2 panels that are To-Do List and Friends next to the calendar page. To-Do List contain timeless events with checkbox so user can always see tasks to perform and when one or more of them are done, user can check it. Other panel is Friends that shows all of users friends with the radio buttons which enable user to select friend to see his unavailable and available times.

5.2.3 Week Appearance of Calendar

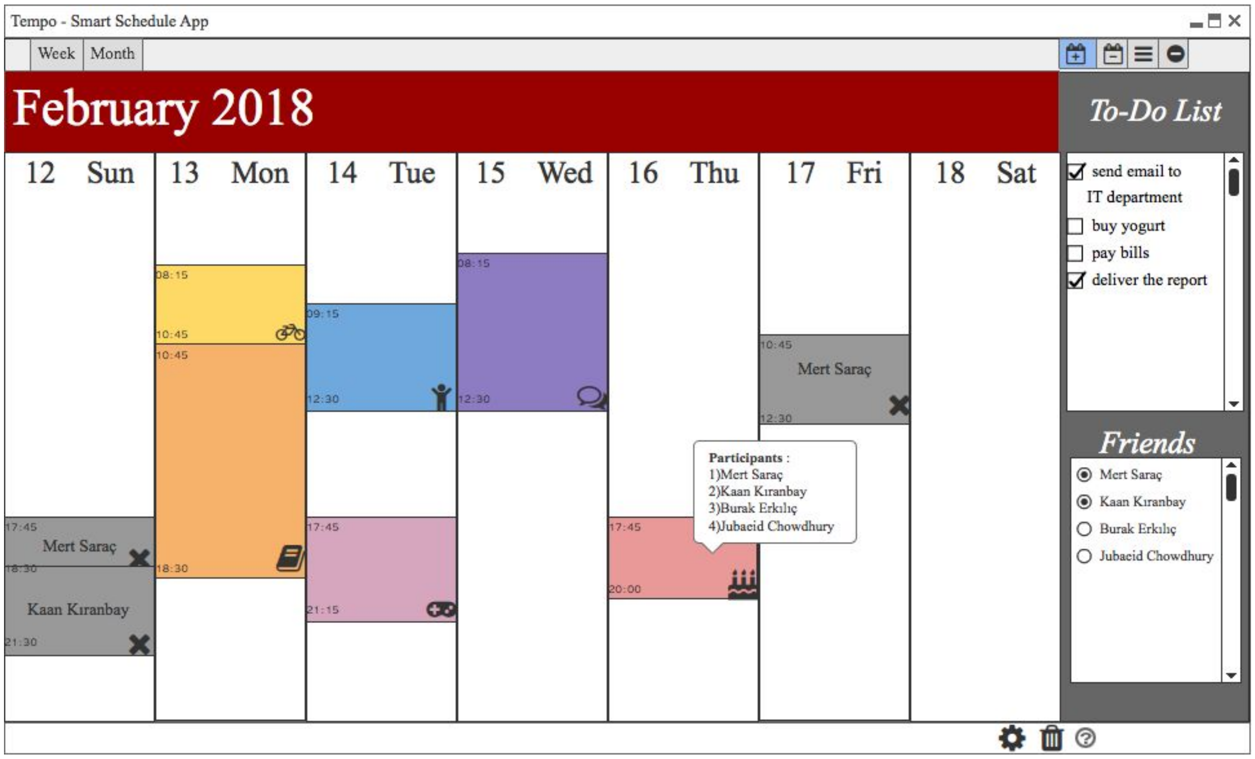
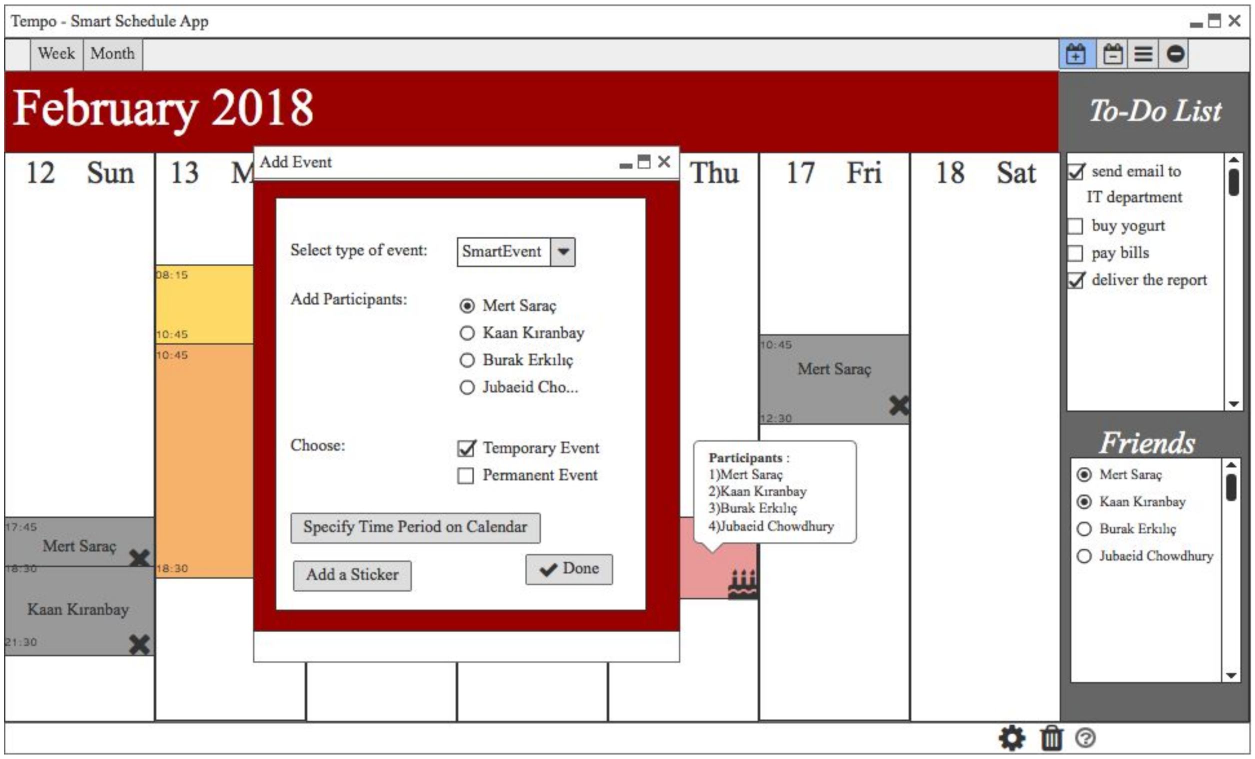


Figure 15 - Calendar Window Week Tab Mock-up

After clicking the "Week" button, user can access this page that contains detailed information about activities like participants of an activity and time period of activity. To see participants, user uses just right click of mouse. To add an activity, user uses add activity button.

5.2.4 Add Event Window



Select type of event: SmartEvent

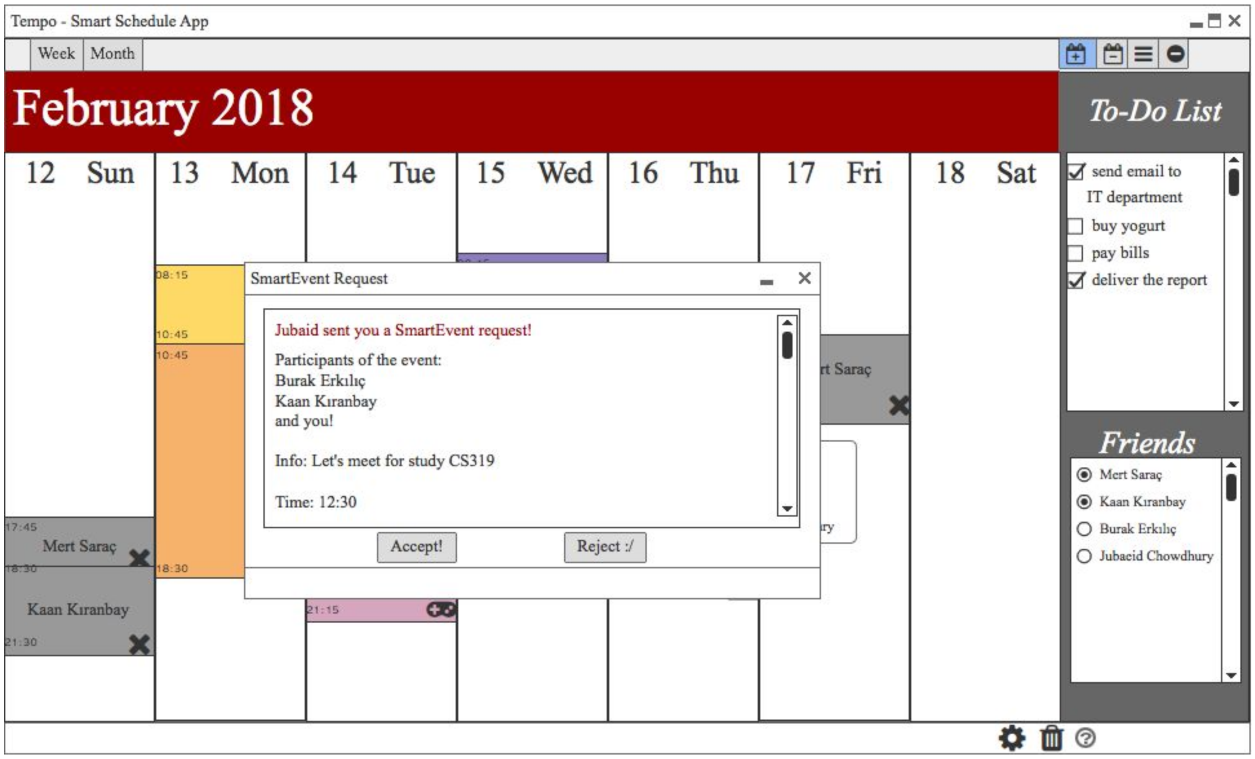
Add Participants: [✓ Mert Saraç] Kaan Kıranbay Burak Erkılıç Jubaeid Cho...

Choose: [✓ Temporary Event] Permanent Event

Figure 16 - Add Event pop-up Window Mock-up

When "Add Event " button is pressed, user encounters with a new window which contains many options for describing the event. User can select whether it is Smart Event or Personal Event, if it is Smart Event, user can select participants otherwise, this part is disabled, and choose whether it is permanent or temporary event. If user wants many stickers are available for event view on calendar. Finally, user needs to specify time period on calendar. If user wants, it is available to add a note to SmartEvent is possible by "Add a Sticker" button.

5.2.5 Notification Window



Jubaid sent you a SmartEvent request!

Participants of the event:
Burak Erkılıç
Kaan Kıranbay
and you!

Info: Let's meet for study CS319

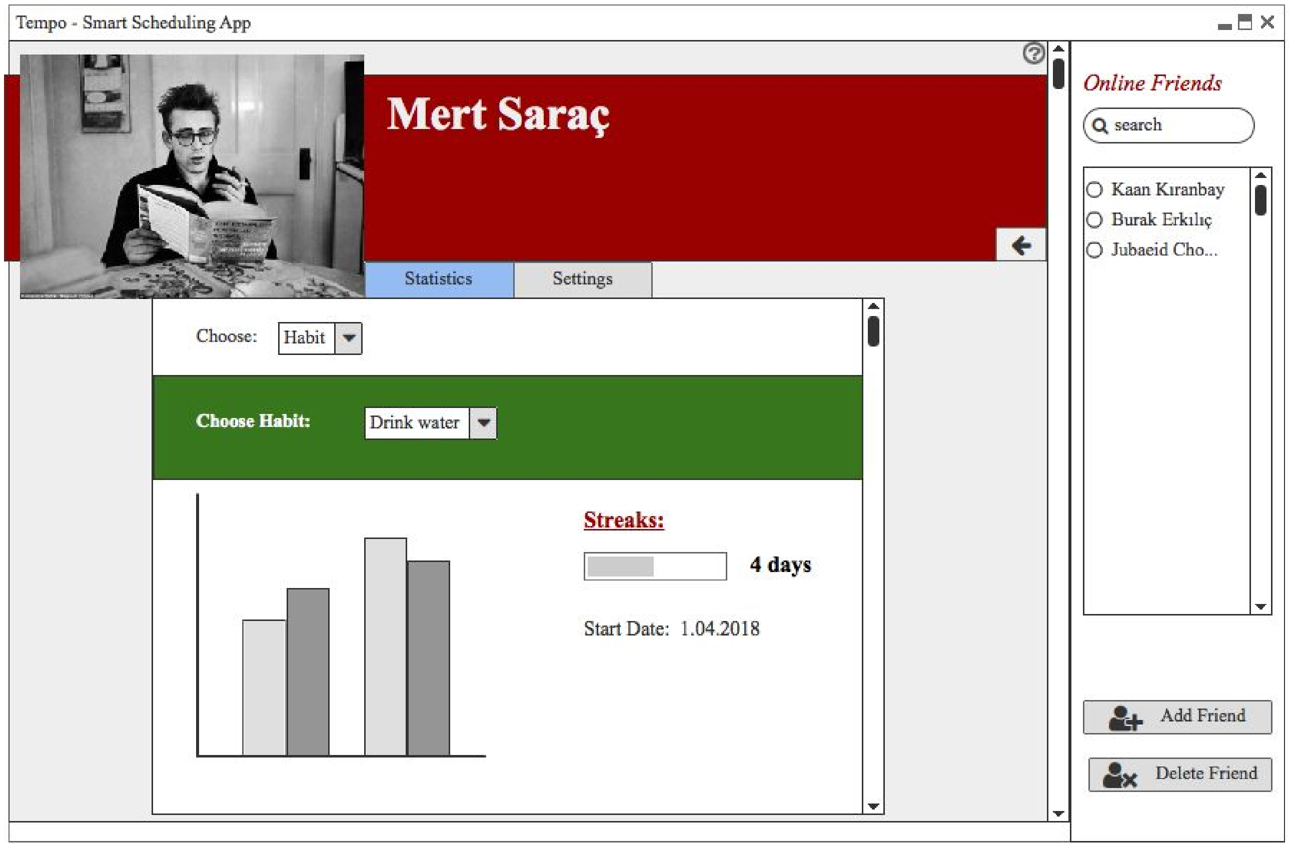
Time: 12:30

- [x] send email to IT department
- [ ] buy yogurt
- [ ] pay bills
- [x] deliver the report

Figure 17 -Notification pop-up Window Mock-up

If one of your friends send you a SmartEvent request, a notification is displayed. This notification contains informations about who sends you the SmartEvent, participants of the SmartEvent, time and explanation about it.

5.2.6 Statistics For Habits



Choose: Habit

Choose Habit: Drink water

Streaks: 4 days

Start Date: 1.04.2018

Figure 18 - Statistics/Settings Window Statistics Tab Habit Mock-up

User can choose whether displaying habits' statistics or tasks' statistics. In Habit's statistic, user can see start date of the habit and streak. Also, there is a detailed chart about the habit is provided. To see another habit's statistics, user should change the habit in the combo box and choose another one.

5.2.7 Statistics For Tasks

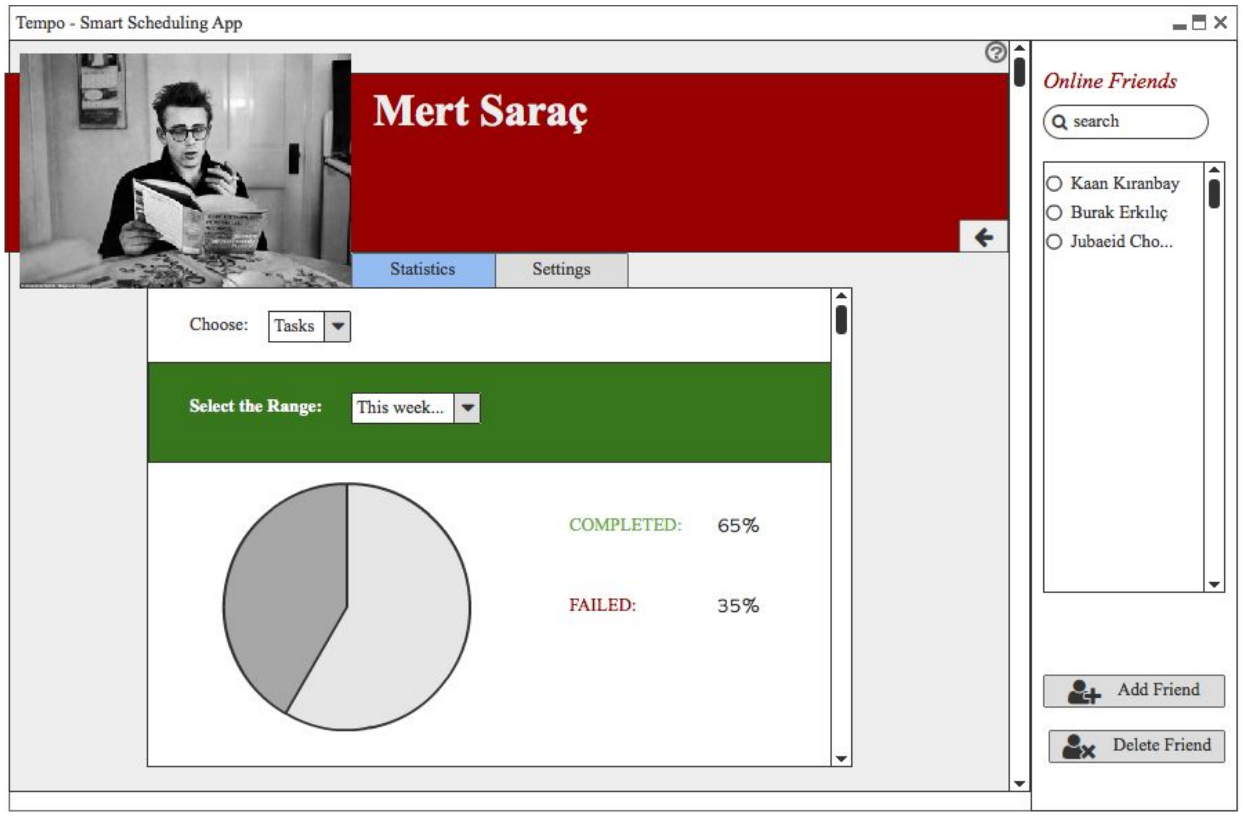
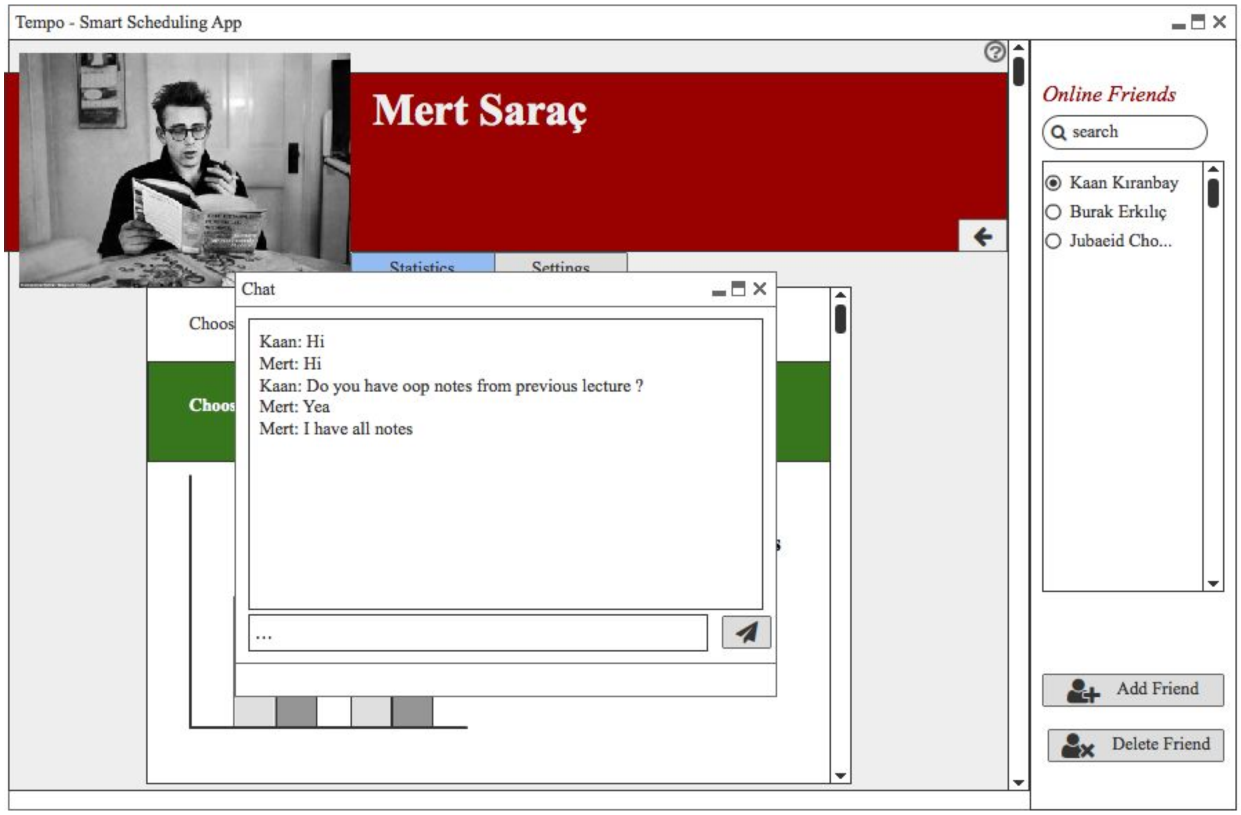


Figure 19 - Statistics/Settings Window Statistics Tab Tasks Mock-up

For tasks, user can see statistics generally but not for every task. In general, user can choose the range such as week, month, year. The percentage of completed and failed is displayed in this page with pie chart about it.

5.2.8 Chat Window



Kaan: Hi
Mert: Hi
Kaan: Do you have oop notes from previous lecture ?
Mert: Yea
Mert: I have all notes

Figure 20 -Chat pop-up Window Mock-up

At the right part of the screen, there is a list of online friends. User can either search for a friend or select one of the online friend to chat. For chat, there is a pop up page.

5.2.9 Add Friend

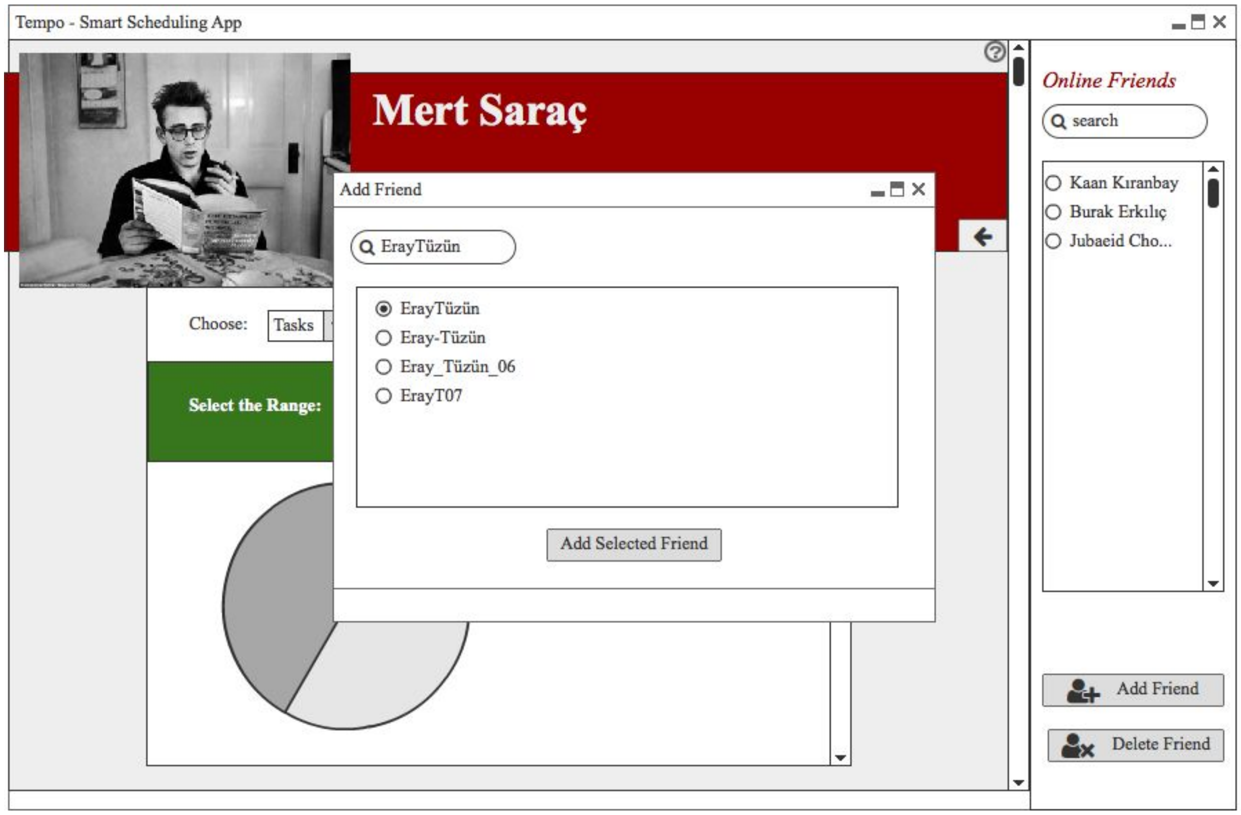


Figure 21 - Add Friend pop-up Window Mock-up

To add a friend, user should click "Add Friend" button. After clicking it, there will be a pop-up page. In this pop-up page, user can type a friend's username and below of it, the results are shown. User click one of the results and click "Add Selected Friend" as a final step.

5.2.10 Delete Friend

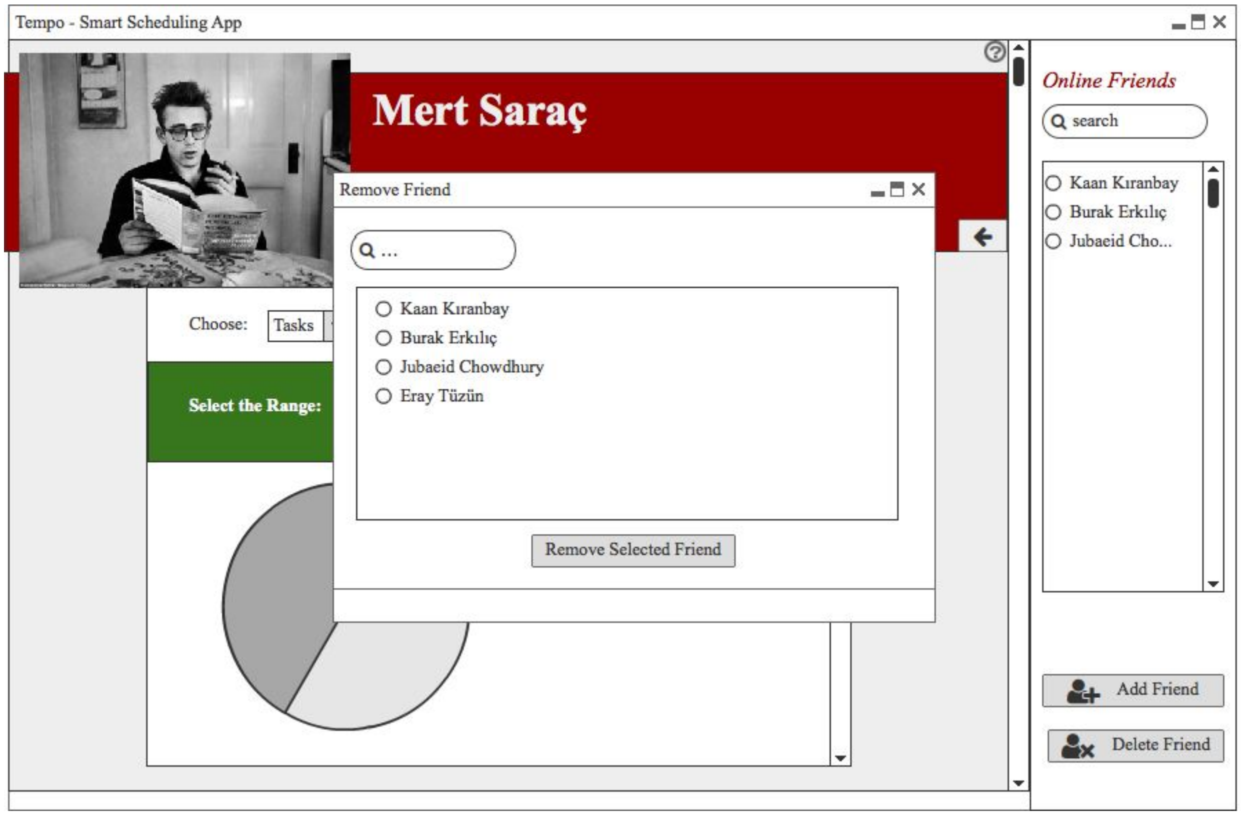
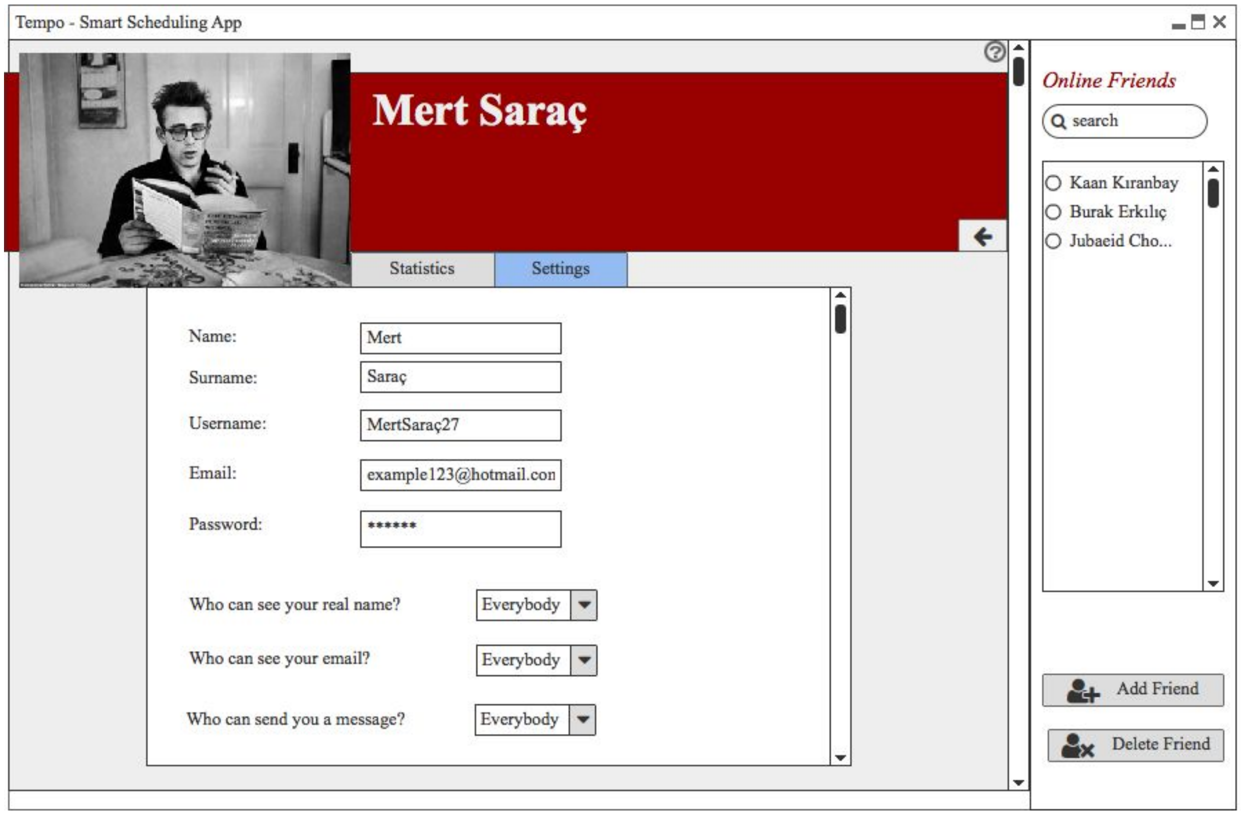


Figure 22 - Remove Friend pop-up Window Mock-up

If user click “Delete Friend” button, a window that contains friend list. Either by using search bar or by selecting from friend list, user choose one to delete and after that, clicks “Remove Selected Friend” as a final step.

5.2.11 Settings



Name: Mert

Surname: Saraç

Username: MertSaraç27

Email: example123@hotmail.con

Password: ******

Who can see your real name? Everybody

Who can see your email? Everybody

Who can send you a message? Everybody

Figure 23 - Statistics/Settings Window Setting Tab Mock-up

In settings tab, there are informations about user and they can be changed or updated. Settings tab also contains some preferences about the program like night mode, language and private related preferences.

5.2.12 Exit Window

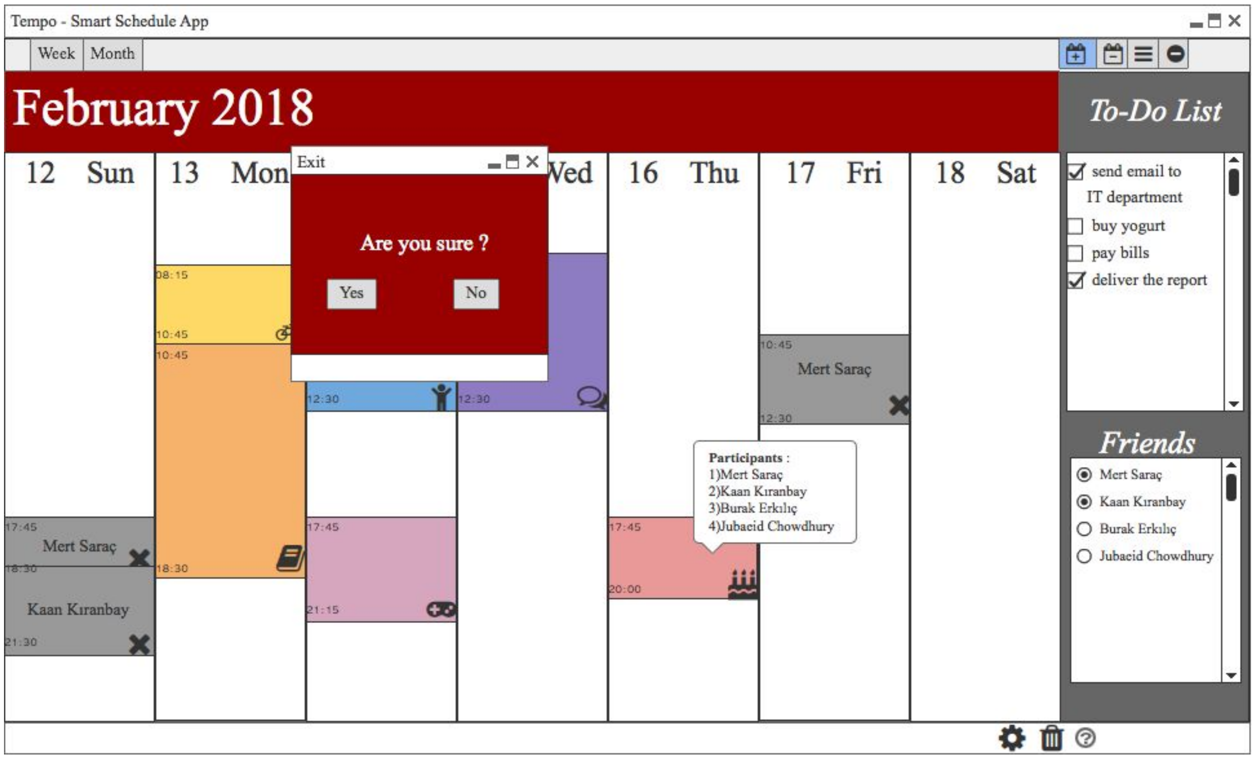


Figure 24 - Exit pop-up Window Mock-up

If user wants to exit, there is a exit button that is next to menu button. After clicking it, a window pops up and asks for it. If user click “Yes”, program is terminated but if “No” button is pressed, main menu shows up again.

6. References

[1] Object-Oriented Software Engineering, Using UML, Patterns, and Java, 2nd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2004, ISBN: 0-13-047110-0.