Honours Year Project Report

TiddlyCalendar

A Collaborative Event Scheduling Calendar Tool

By

Lin Liansen, Michael

Department of Information Systems

School of Computing

National University of Singapore

2007/2008

Honours Year Project Report

TiddlyCalendar

A Collaborative Event Scheduling

Calendar Tool

By

Lin Liansen, Michael

Department of Information Systems

School of Computing

National University of Singapore

2007/2008

Project No: H041160

Advisor: Associate Professor Martin Henz

Deliverables:

Report:	1 Volume
Program:	1 CD-ROM

Abstract

TiddlyCalendar is a PIM calendar application developed entirely in Javascript, DHTML and CSS. Unlike traditional Javascript calendar which mainly displays monthly calendar information (much as physical calendar) or is used to select dates (date-pickers in forms), TiddlyCalendar offers features and functions that is comparable to desktop and server calendar applications such as Microsoft Outlook and Google Calendar, allowing users to manage their schedules and appointments through the TiddlyCard platform, which provides a workspace for collaborative work.

TiddlyCalendar has introduced innovative features in arranging common appointments. The main highlight is the Appointment Polling system which solves a long-standing stagnant problem in current calendar applications when arranging for a common appointment/meeting date/time. Its Public Appointment Sharing feature allows easy notifications to changes to shared appointments to all users, eliminating unnecessary steps such as repeated export/import of appointment. These features increase efficiency and effectiveness, especially when collaborating with multiple parties.

Subject Descriptors:

D.2.13 Reusable Software	*Reusable libraries*
H.3.5 Online Information Services	*Data sharing*
H.4.1 Office Automation	*Time management*
H.5.3 Group and Organization Interfaces	*Computer-supported cooperative work*
	Web-based interaction

Keywords:

Calendar, Schedule, Time Management, Collaboration, Personal Information Management (PIM)

Implementation Software:

Javascript, DOM, DHTML, CSS, web-browsers

Acknowledgement

I would like to extend my thanks and gratitude to my project supervisor Associate Professor Martin Henz for his patience, guidance, advice and support throughout the entire period of this project.

I would like to extend my thanks to Mr Melvin Zhang for his comments and suggestions and the TiddlyCard Developer Community for their effort and assistance, in particular:

a.	Mr. Daryl Seah	for providing backend support & infrastructure
b.	Mr. Lek Hsiang Hui	for implementing modified Comet architecture
c.	Mr. Siva Subramanian	for integration with Asalta
d.	Mr. Tang Tung Leh	for feedback and suggestion to TiddlyCalendar
e.	Mr. Tran Khoa Nguyen	for audio support for TiddlyCalendar Alarm
f.	Ms. Wang Shangshang	for integration with TiddlyRSS

List of Figures

Table of Contents

1. Introduction

1.1 TiddlyCard

1.1.1 Overview of TiddlyCard

TiddlyCard is a browser-based platform that is written entirely in Javascript. Inspired by HyperCard (HyperCard, 2007), TiddlyCard aims to provide a cross-browser environment for rapid application development, enabling developers to create and deploy applications that are client-centric, server supported and platform independent.

From standalone applications to the rapid adoption of the web, the software industry has seen a change in choice of medium preferred by users. This phenomenon has continued to evolve and a new trend has emerged with technologies such as Adobe AIR (Adobe Labs, 2006) slated for release in the near future (currently in beta 2). This trend is a leap in making applications available to users, providing continuous functionality and operability regardless of time, platform and device.

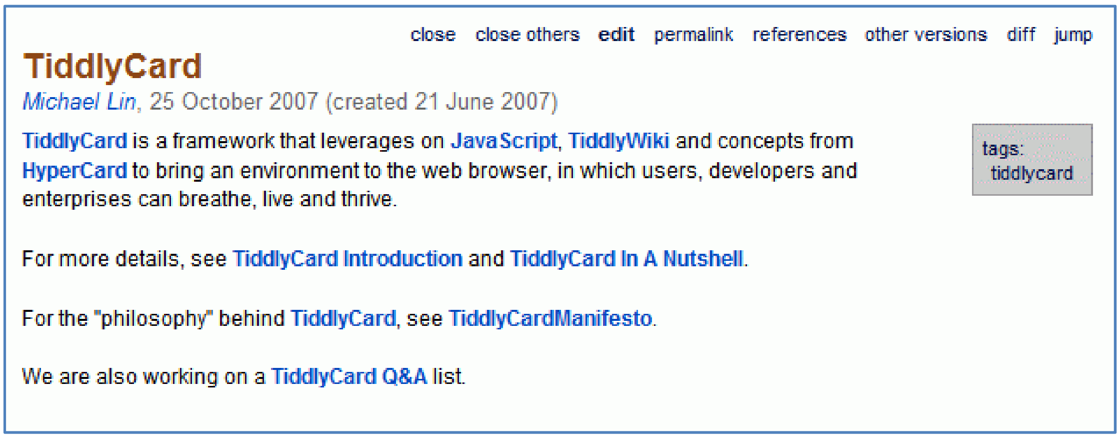
Unlike traditional web-based applications which require uninterrupted server connectivity, TiddlyCard applications are client-centred and remain functional with or without connectivity, enabling users to use the applications anytime without disruption. When a network connection is available, TiddlyCard applications may make use of the server capabilities to perform tasks such as synchronisation of content repository (automatically or at user's discretion) and collaboration.

Built entirely on Javascript and a browser-based framework, TiddlyCard is platform independent, enabling users to access TiddlyCard applications through desktop computers, laptops and any browser-enabled devices including Personal Digital Assistants (PDAs) and mobile phones regardless of the underlying operating system.

1.1.2 Foundation of TiddlyCard

TiddlyCard is an extension of the increasingly popular TiddlyWiki (Ruston, 2004), a free single-page application that is written entirely in Javascript, HTML and CSS. It allows anyone to create personal self-contained hypertext documents that can be posted to a web-server, sent via email or kept on a thumbdrive.

TiddlyWiki makes use of micro-content known as '**tiddlers**' (Figure 1) to store small fragments of content which are linked using wiki style and referenced through hyperlinks, which users can click to progressively display the content although the page is loaded all at once.


close close others edit permalink references other versions diff jump
tags: tiddlycard

TiddlyCard

Michael Lin, 25 October 2007 (created 21 June 2007)

TiddlyCard is a framework that leverages on JavaScript, TiddlyWiki and concepts from HyperCard to bring an environment to the web browser, in which users, developers and enterprises can breathe, live and thrive.

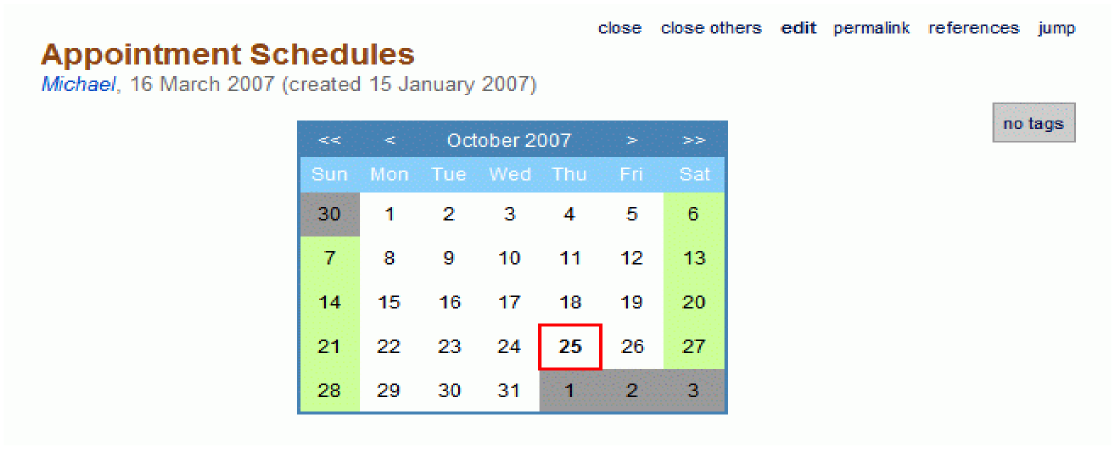
For more details, see TiddlyCard Introduction and TiddlyCard In A Nutshell.

For the "philosophy" behind TiddlyCard, see TiddlyCardManifesto.

We are also working on a TiddlyCard Q&A list.

Figure 1: Tiddler (View Mode)

TiddlyWiki also allows the creation of special tiddlers known as '**plugins**' and '**macros'** to extend the capabilities of TiddlyWiki. Plugins can contain Javascript codes which can be written by the user/developer or pre-defined Javascript functions provided by the TiddlyWiki API Library while macros execute the code in plugins and display the resulting behaviours. An example of a macro is the PlasticCalendar (Soares, 2006) which is used to create the calendar shown below (Figure 2).



Appointment Schedules

Michael, 16 March 2007 (created 15 January 2007)

<<	<	October 2007			>	>>
Sun	Mon	Tue	Wed	Thu	Fri	Sat
30	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3

Figure 2: PlasticCalendar Macro

1.1.3 Objectives of TiddlyCard

The main objectives of TiddlyCard are:

a. Platform independence – Provides freedom of choice of operating systems and browser
b. Ease of configurability – Provides ability to customise features and integrate components easily
c. User empowerment – Provides capability to create, control and manipulate data without reliance on application or content providers
d. Security – Provides protection of privacy and reduction of vulnerabilities to malicious users
e. Collaborative Workspace – Provides a suite of applications to assist in collaboration tasks to facilitate communication, content creation and sharing

1.1.4 Components of TiddlyCard

Components of TiddlyCard include:

a. Asalta – a WYSIWYG designing tool
b. Debugger – a debugging tool for TiddlyCard
c. Form Generation and Handling – a tool for automatic form generation
d. Messenger – an Instant Messaging tool for TiddlyCard
e. Multimedia Resource Manager – a multimedia player
f. Plugin System – a framework to support TiddlyCard plugins
g. TiddlyCalendar – a collaborative event scheduling calendar tool
h. TiddlyRSS – a RSS reader
i. Version Control and Server-Side Support – backend supporting framework

1.2 The Calendar System

1.2.1 Overview of Calendar

A calendar is a system for keeping track of time using units of days, weeks, months and years. These divisions of time are based on the movements of the earth, the moon and the sun.

A day is the amount of time it takes for the sun to rise, set and rise again. This is determined by the rotation of the earth on its axis and the position of the sun. The sequence of 7 days (Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, and Saturday) is grouped into units of weeks (please refer to Table A1 in Appendix A for the origins of English week day names). Where the norm start day is Sunday, ISO 8601 Week Date (Nen, 2000) specifies Monday as the first day of the week and Sunday as the last. (Santino, 1994)

A month is the amount of time it took for the moon to complete its cycle of phases – from new moon to full moon and back to new moon. The word month comes from moon. The origins of English naming for calendar months can be found in Table A2 in Appendix A. (Santino, 1994)

A year is the amount of time it takes the earth to complete one revolution around the sun. It is determined by astronomers to be 365 days, 5 hours, 48 minutes and 46 seconds. (Santino, 1994)

1.2.2 History of Solar Calendars

The Roman calendar, Julian calendar and Gregorian calendar are the three most significant in the history of solar calendars.

The ancient Roman calendar initially had only 10 months and the year began in March (when farmers began work for growing season) before it was changed to a 12-month system with January as the beginning of the year. It consisted of 355 days with March, May, July and October with 31 days, while February has 28 days and the rest with 29 days. This year did not fit with the actual solar year, leading to an extra period (known as Intercalaris) of 22-23 days being added to make the calendar more correct. However, over time, this calendar became wildly out of sequence. (Santino, 1994)

In 46 B.C, Julian Caesar introduced a new calendar that had 365 days, with one day added every fourth year (leap year). He distributed the additional 10 days among the 29-day months, establishing the calendar that is identical to todays. This calendar is named Julian calendar after its inventor. (University of Chicago, 1993)

Conversely, this correction of one day in every four years made the calendar year longer than the year of the seasons and in 1582, the vernal equinox (beginning of spring) occurred on March 11 instead of the correct date March 21. (University of Chicago, 1993)

Pope Gregory XIII dropped 10 days from the calendar in 1582 (October 4, 1582 is followed by October 15, 1582) to make it correspond more closely to the seasons

(University of Chicago, 1993). He also directed that leap years that occur in century years be omitted unless it is divisible by 400 i.e. omit 3 leap years every 400 years. (Santino, 1994). This calendar became known as the Gregorian calendar. Although it did not gained immediate acceptance, it was gradually accepted over 3 centuries and became the most common form of calendar in use today.

1.2.3 Calendar Usage and Revolution

Traditional calendar has been used primarily as a tool mainly to identify days to inform about or agree on a future event and record an event that has happened. Its significance lies in the ability to depict days of importance in civil, religious and social contexts. For example, a calendar provides a way to determine which days are civil and religious holidays, which days mark the beginning and end of business accounting periods and which days have legal significance such as contract expiry date.

Over the years, the calendar gradually evolved from merely informing dates to become part of the Personal Information Management (PIM) application: a tool that helps people to manage their personal schedule, time and activities. Users are able to refer to their calendar to check time availability prior to confirmation of an appointment and be reminded about upcoming events amidst their busy schedule so that tasks are not missed and appointments are not forgotten.

With rapid globalisation and the need for collaboration among individuals, the calendar takes a stride in another new direction with calendar sharing. While calendar applications developed for the personal computer uses the internet and web technologies such as email as a channel for users to share their calendar information through the use of standardised file formats such as iCalendar (iCal), XML documents and even CSV text files, web calendars such as Google Calendar provided its users a more convenient and integrated method of event tracking and notification through the use of public shared calendars that is available online.

2. TiddlyCalendar

2.1 Overview of TiddlyCalendar

TiddlyCalendar is a PIM application developed entirely in DHTML, Javascript and CSS for TiddlyCard. TiddlyCalendar aims to bring TiddlyCard one step closer to fulfilling its objective of creating a collaborative workspace by allowing users to create and share common appointments, meetings and events while maintaining their own personal schedule of events for easy referencing.

Initially developed as a plugin for TiddlyWiki, TiddlyCalendar has evolved from a prototype similar to PlasticCalendar that is limited in capabilities to become a near full-fledged calendar application that is rewriting a new chapter in Javascript calendars. TiddlyCalendar has features comparable to that of desktop calendar applications such as Rainlendar (Rainy, 2007) and Microsoft Outlook in contrast to the conventional calendar display and date-picker found in Javascript calendars.

In addition, TiddlyCalendar retains the benefits provided by TiddlyWiki which allows linkages to existing TiddlyCard content through wiki links, allowing calendar events to be linked to tiddler content which may, for example, contain meeting agenda and minutes to provide a more comprehensive picture of the event instead of just presenting the date and time.

2.2 Motivation of TiddlyCalendar

2.2.1 Public Shared Appointment Importing/Exporting

Although the calendar has taken giant leaps; from a physical piece of paper to become a PIM application and eventually, sharing of calendars over the web, the current method of calendar sharing still presents inconvenience and hassle to the user. This can be illustrated with the existing way to perform calendar sharing.

Current calendar sharing is based on importing and exporting calendar events in standardised file formats such as iCal, XML documents and even CSV files. The following steps are performed in sequence:

1. User A creates/edits a public appointment

2. User A exports the calendar file and uploads it onto a web server or ‘publish’ the calendar to a WebDav-enabled server (Whitehead, 2007) which supports the CalDav protocol (Dusseault, 2005)
3. User B can either download the iCal file onto local computer and imports the events into his/her calendar (method A) or subscribe to the published calendar (method B)

This method of calendar sharing (import/export mechanism) presents a number of difficulties and inconvenience:

Method A

1. The entire export (user A) and import (user B) process has to be repeated for every single public appointment that has changed
2. User A has to notify user B explicitly about the calendar change and request for user B to import the updated calendar file
3. Information (e.g. additional details) may be lost when user B overwrites his existing local copy of the appointment with the imported appointment information

Method B

4. Subscribed calendars provide a single centralised copy of the appointments which are updated automatically by the calendar client. However, subscribed appointments are read-only and does not allow any form of editing

The above gets more cumbersome as more parties are involved e.g. if 5 people are interested in each other’s activities, they will have to repeat the entire export/import process 4 times (1 time for each different user).

In addition, changes to a public shared event often depended on a single user to perform the update. Additional details/information either cannot be added or edited by other parties (subscribed calendars) or requires the repeat of the entire import/export process and notifications sent to all relevant parties. As there is no central single source (method A), the calendar’s integrity is compromised when multiple parties update the same appointment and send out notifications simultaneously.

2.2.2 Common Appointment Scheduling Among Multiple Parties

Multiple users are often perplexed and frustrated when trying to find a suitable date/time for an appointment e.g. a group of students deciding on a meeting date/time using the process below:

a. Meeting coordinator proposes an initial date/time for meeting and sends notification (email, SMS, etc) to all team mates to request for confirmation of attendance
b. Meeting coordinator waits for responses
c. Team-mates receive notification and inform meeting coordinator of attendance
d. Meeting coordinator collates responses and inform the result (meeting confirmed, reschedule, etc) to everyone

In the event that the date/time cannot be agreed upon, the process is repeated. This process imposes a toll on the meeting coordinator and the process creates unnecessary delays (mainly from collating of results and repetition of the scheduling process) before the finalisation of the appointment date/time.

Not only does it take long to finalise, the above presents another problem. If the process is carried out over other channels such as email and SMS, team-mates may forget about the appointment and appointment details are prone to human errors (e.g. enter wrong date/time, incorrect venue, etc). The process is highly inefficient and ineffective.

2.2.3 Integration

Current applications are often created as standalone applications with limited integration capabilities. This holds true especially for calendar-related applications and usage. This can be illustrated by an example of a meeting. Prior to meeting, date/time slot for the meeting is often discussed via channels such as emails, SMS or Instant Messaging programs. The agenda of the meeting is often created as a text document and sent to the relevant parties. After the meeting, the minutes are distributed in the same manner as well.

In the above example, 3 different applications are required; a word processing application (for creating agenda and minutes document), an email client or instant messaging program (for sending notification and arranging the date/time of meeting) and a calendar application (keep track of appointment details and provide reminder/alert before the meeting).

These applications provide no way to reference the content of each other easily, resulting in users having to revert to different documents and programs for related information e.g. when user is reminded of meeting in the calendar application, he has to open the agenda document in a separate word processing application.

Besides this, the information that is stored in a calendar application cannot be readily accessed by other applications (e.g. meeting date/time must be re-entered into the agenda document using plain text).

2.2.4 Requirement of TiddlyCard

TiddlyCard's goal is to provide a common workspace for users to collaborate effectively using one single platform. TiddlyCalendar provides an event scheduling tool for users to keep track of appointments that are relevant to the collaboration effort such as meeting appointments and project deadlines. In addition, it also allows the user to keep track of their personal events in one single calendar to better manage their schedule and arrange appointment dates and times.

2.3 Components of TiddlyCalendar

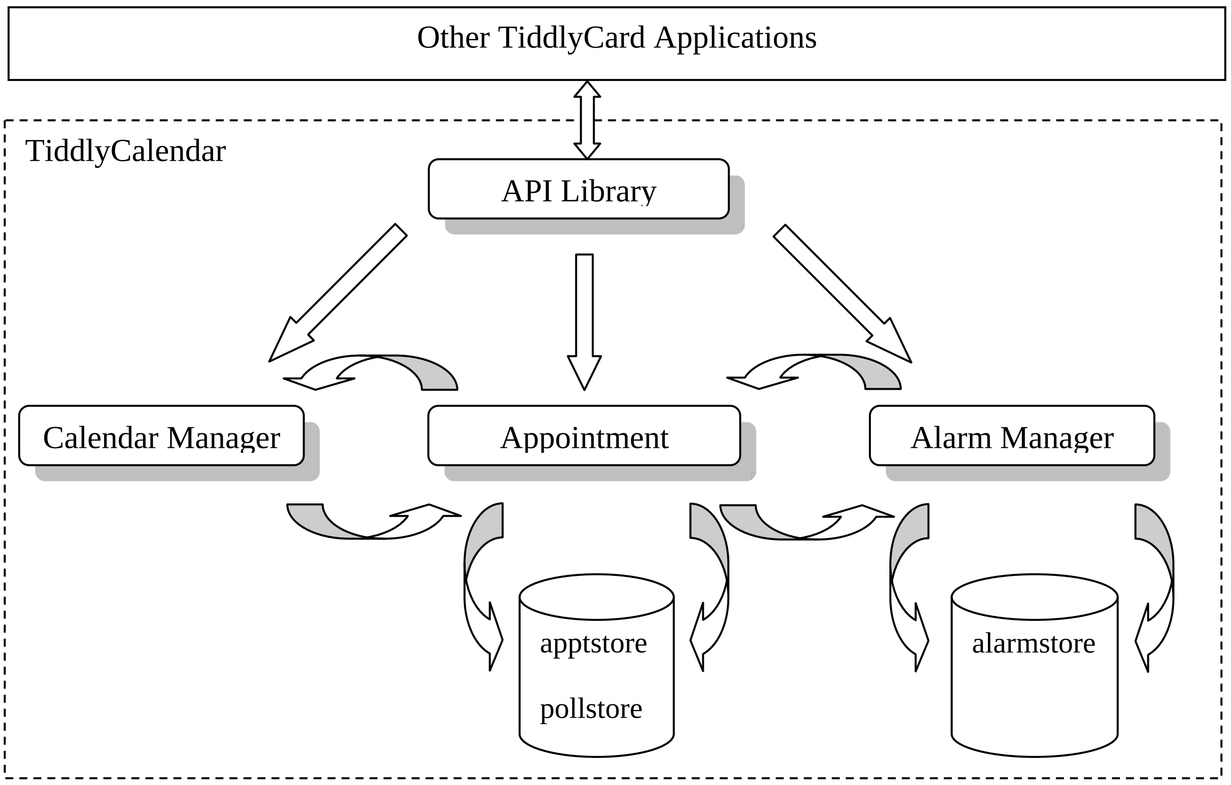


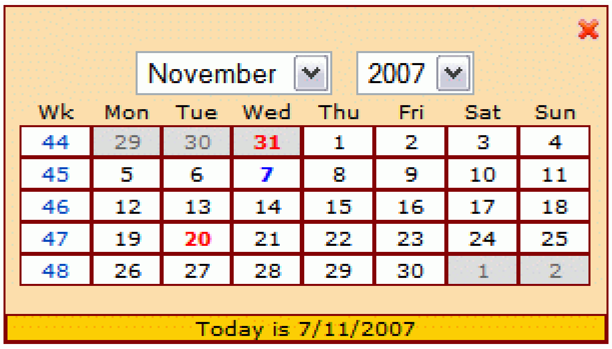
Figure 3: Components of TiddlyCalendar

TiddlyCalendar consists of 4 main components, namely the Calendar Manager, the Appointment Manager, the Alarm Manager, the Datastores and the TiddlyCalendar API Library (Figure 3).

2.3.1 Calendar Manager

The Calendar Manager is responsible for displaying information on appointments to the user. It consists of:

a. Event List window which shows a summarised list of upcoming appointments for the following 14 days from today's date
b. Calendar window (Figure 4) which highlights the dates with appointments for the selected month/year (see Section 4.1.5)



Wk	Mon	Tue	Wed	Thu	Fri	Sat	Sun
44	29	30	31	1	2	3	4
45	5	6	7	8	9	10	11
46	12	13	14	15	16	17	18
47	19	20	21	22	23	24	25
48	26	27	28	29	30	1	2

Today is 7/11/2007

Figure 4: Calendar Window

c. Week View (Figure 5) window which displays the list of appointments for the selected week (see Section 4.1.5)

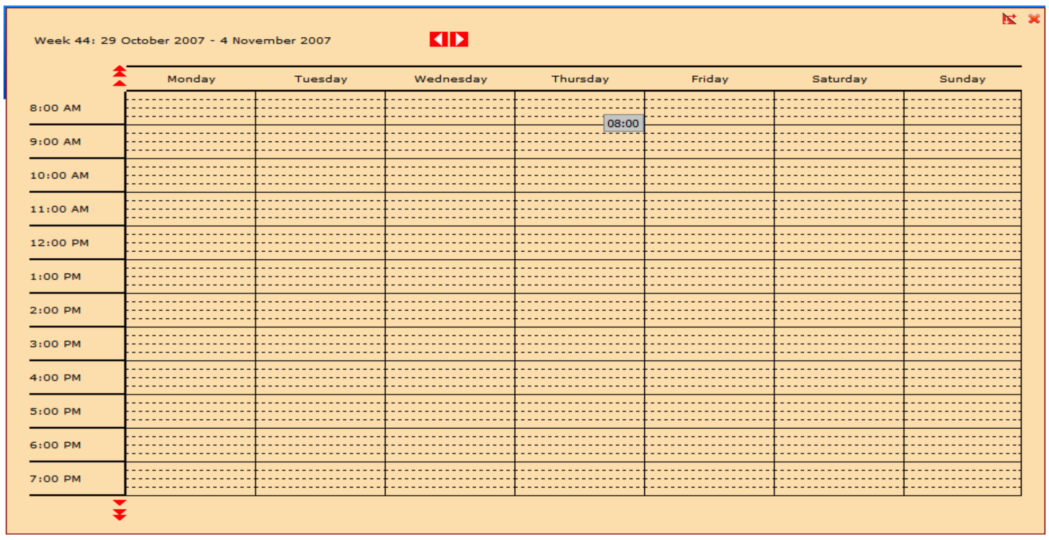

Week 44: 29 October 2007 - 4 November 2007

Figure 5: Week View Window

d. Summary window which displays summary of appointments for a date
e. Detail window which displays details of selected appointment

2.3.2 Appointment Manager

The Appointment Manager performs all tasks/operations related to appointment. Such tasks/operations include:

a. Add/Edit/Delete Appointment
b. Search Appointment
c. Load/Save Appointments
d. Public Events Sharing (see Section 3.1)
e. Event Polling (see Section 3.2)
f. Import/Export Appointments (see Section 3.4)

2.3.3 Alarm Manager

The Alarm Manager keeps track of reminders and alerts for appointments. It displays notification of upcoming appointments and can be run independently from Calendar Manager and Appointment Manager. This allows the user to continue to receive notification even if TiddlyCalendar is not fully loaded with all the components.

2.3.4 API Library

The API Library provides a suite of commands which can be executed by external TiddlyCard applications to perform calendar-related tasks such as showing/hiding of calendar window to check schedule from another application. This interface encapsulates the different functions of TiddlyCalendar, making changes invisible to the external application and ensuring continuous functioning of calendar operations from external applications without the need for external application developer to modify any code.

3. Innovation and Highlights

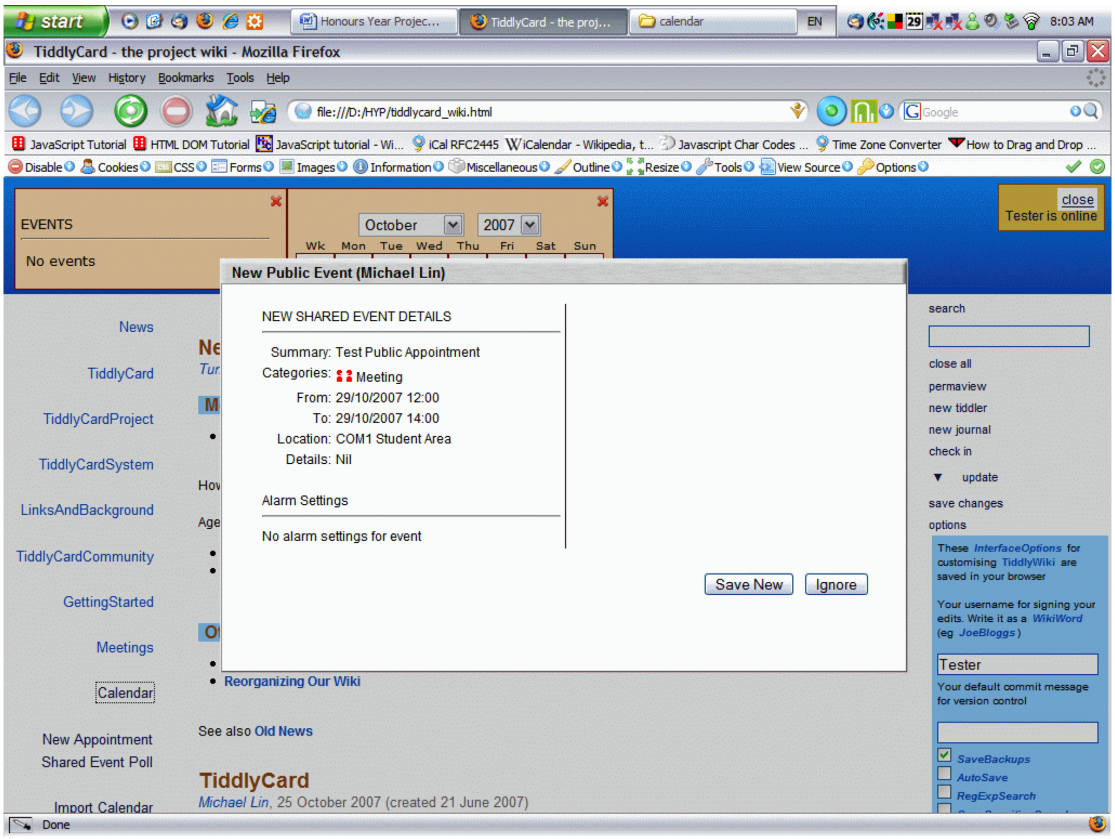
3.1 Public Appointment Sharing

In order to counter the problem of having a coordinator to update and reflect all changes to an appointment, TiddlyCalendar introduced a new system of public event sharing. Traditionally, any changes to an appointment involve the exporting of the newly updated calendar and the importing of the new appointment by the various parties. Although this situation can be avoided through the use of calendar subscription, this solution is rigid as it does not allow the users to edit any details due to its read-only nature.

TiddlyCalendar's new public appointment sharing system resolves this issue through the use of a peer distribution system. Through this system, each shared public appointment is cloned on each individual's calendar, granting every user the ability to control and coordinate the calendar appointment. Changes to the calendar appointment can be performed by anyone instead of having to rely on a coordinator.

Appointments are communicated through the peer distribution system instantaneously. Using a modified version of the Comet (see Section 4.1.3) architecture (Comet (programming), 2007), TiddlyCalendar allows users (who are previously offline) to perform synchronisation of calendar appointments which can be stored, retrieved and updated much like MSN offline messages (Figure 6). This eliminates the need to export/import calendar files, providing a more efficient way of public appointment sharing.

Unlike the export/import mechanism where edited calendar appointments directly overwrites the existing calendar appointment (e.g. Rainlendar) or create a new public appointment instead of updated the existing calendar appointment (e.g. Microsoft Outlook 2007), TiddlyCalendar provides a control mechanism which allows the user to specify the action when such an update occurs (Figure 7). When a public appointment is updated, the user is able to compare the differences between the updated appointment and the existing copy before deciding whether to accept the changes or preserve the existing copy, thereby ignoring the change and saving the appointment as a private appointment where additional private details can be added.



New Public Event (Michael Lin)

NEW SHARED EVENT DETAILS

Summary: Test Public Appointment
Categories: Meeting
From: 29/10/2007 12:00
To: 29/10/2007 14:00
Location: COM1 Student Area
Details: Nil

Alarm Settings

No alarm settings for event

These InterfaceOptions for customising TiddlyWiki are saved in your browser

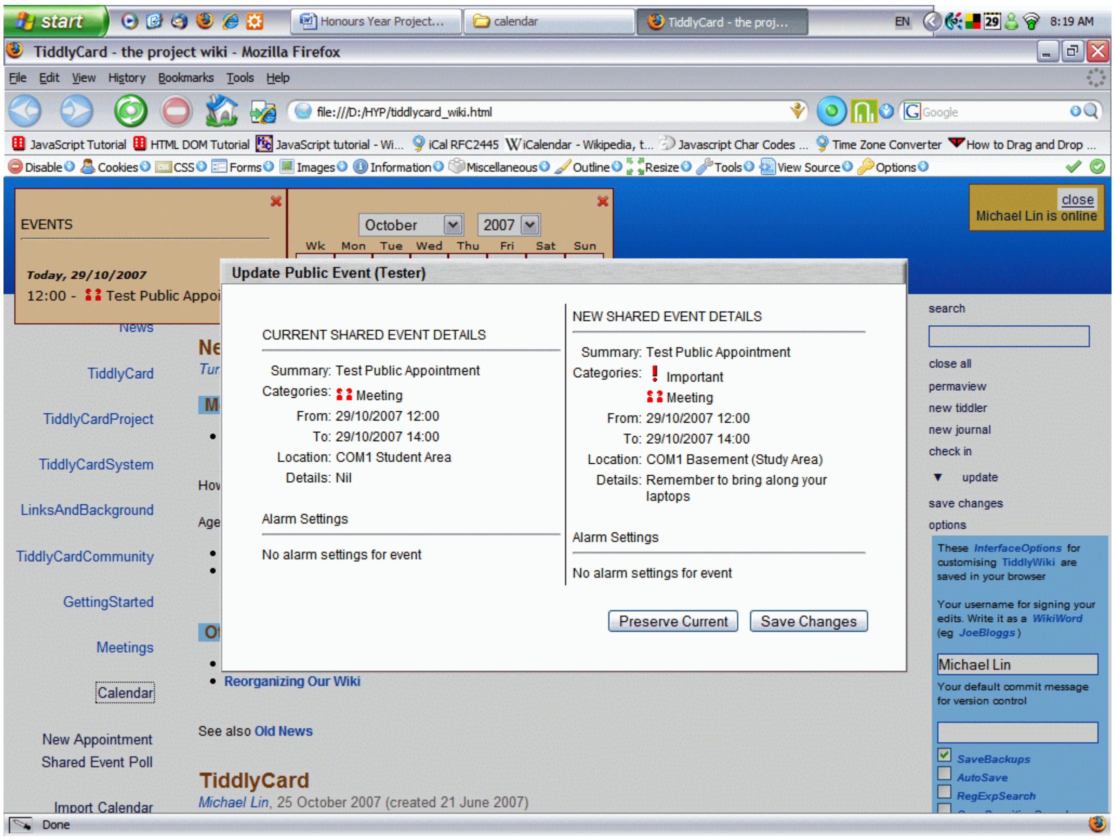
Your username for signing your edits. Write it as a WikiWord (eg JoeBloggs)

Tester

Your default commit message for version control

☑ SaveBackups
☐ AutoSave
☐ RegExpSearch

Figure 6: Instantaneous Sharing with Synchronisation (when connectivity is available)



Today, 29/10/2007
12:00 - Test Public Appoi

Update Public Event (Tester)

CURRENT SHARED EVENT DETAILS

Summary: Test Public Appointment
Categories: Meeting
From: 29/10/2007 12:00
To: 29/10/2007 14:00
Location: COM1 Student Area
Details: Nil

Alarm Settings

No alarm settings for event

NEW SHARED EVENT DETAILS

Summary: Test Public Appointment
Categories: Important
Meeting
From: 29/10/2007 12:00
To: 29/10/2007 14:00
Location: COM1 Basement (Study Area)
Details: Remember to bring along your laptops

Alarm Settings

No alarm settings for event

These InterfaceOptions for customising TiddlyWiki are saved in your browser

Your username for signing your edits. Write it as a WikiWord (eg JoeBloggs)

Michael Lin

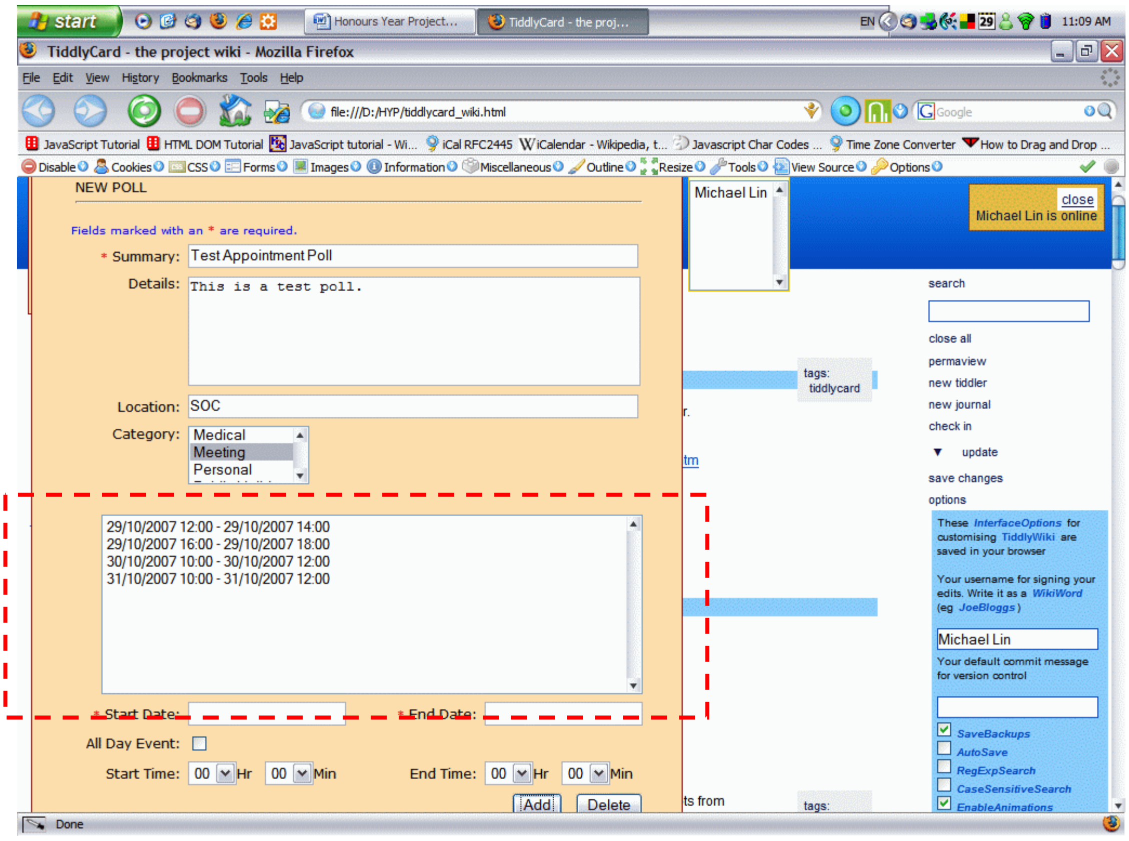
Your default commit message for version control

☑ SaveBackups
☐ AutoSave
☐ RegExpSearch

Figure 7: TiddlyCalendar Update Control Mechanism

3.2 Appointment Polling

TiddlyCalendar has initiated a new polling system that is built upon Doodle (Inturico Engineering, 2006) in calendar applications. Contrary to the conventional method of fixing an appointment through repetitive request for date/time slot in conflicting schedule situations, TiddlyCalendar's polling system allows the appointment coordinator to specify a number of date/time slots when creating the appointment poll (Figure 8), thereby allowing the participants to select their desired slots from those proposed.



NEW POLL

Fields marked with an * are required.

* Summary:	Test Appointment Poll
Details:	This is a test poll.
Location:	SOC
Category:	Medical / [✓ Meeting] / Personal

29/10/2007 12:00 - 29/10/2007 14:00
29/10/2007 16:00 - 29/10/2007 18:00
30/10/2007 10:00 - 30/10/2007 12:00
31/10/2007 10:00 - 31/10/2007 12:00

* Start Date: * End Date:
All Day Event: ☐
Start Time: 00 Hr 00 Min End Time: 00 Hr 00 Min
Add Delete

These InterfaceOptions for customising TiddlyWiki are saved in your browser

Your username for signing your edits. Write it as a WikiWord (eg JoeBloggs)

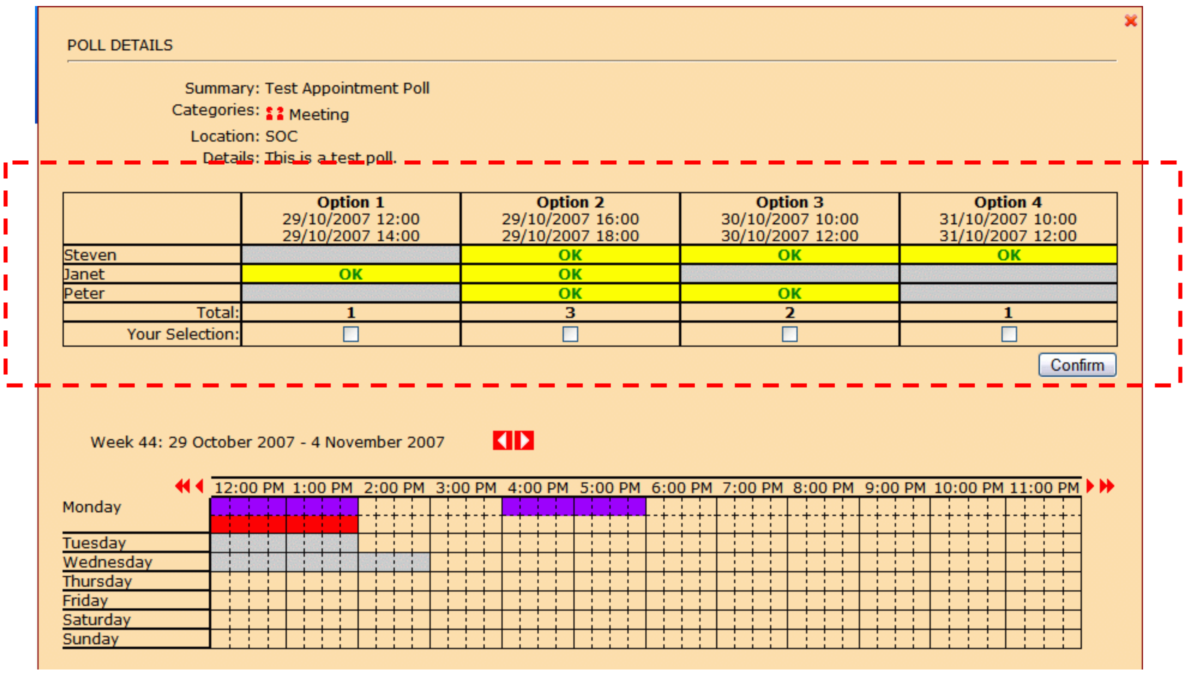
Michael Lin

Your default commit message for version control

- [✓] SaveBackups
- [ ] AutoSave
- [ ] RegExpSearch
- [ ] CaseSensitiveSearch
- [✓] EnableAnimations

Figure 8: Multiple Date/Time Slots for Event Polling

When participants receive the appointment poll, they are presented with information about the appointment details such as Summary, Location, Details and Category as well as a list of responses (collated automatically by TiddlyCalendar) from other participants who have voted for their desired slots (Figure 9). This collated list may be used by the appointment coordinator to make decision on the final selected date/time slot based on user-specified criteria e.g. based on majority attendance or attendance of specific individual(s) as priority.



POLL DETAILS

Summary: Test Appointment Poll
Categories: Meeting
Location: SOC
Details: This is a test poll.

	Option 1 29/10/2007 12:00 29/10/2007 14:00	Option 2 29/10/2007 16:00 29/10/2007 18:00	Option 3 30/10/2007 10:00 30/10/2007 12:00	Option 4 31/10/2007 10:00 31/10/2007 12:00
Steven		OK	OK	OK
Janet	OK	OK		
Peter		OK	OK	
Total:	1	3	2	1
Your Selection:	☐	☐	☐	☐

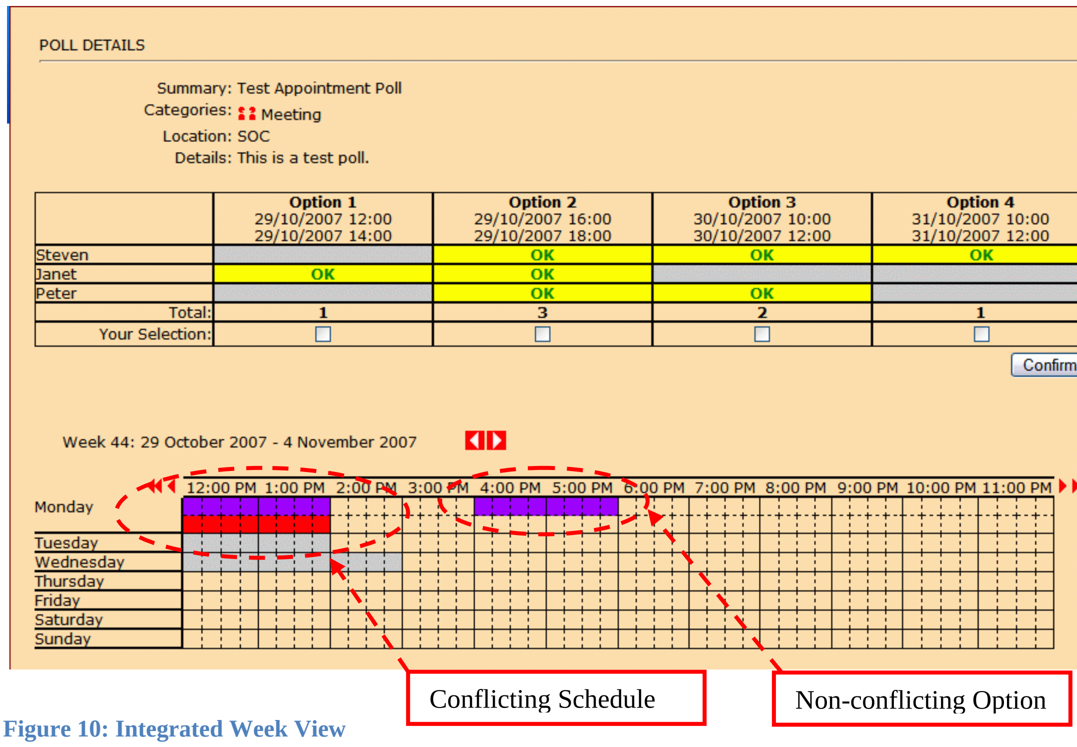
Confirm

Week 44: 29 October 2007 - 4 November 2007

	12:00 PM	1:00 PM	2:00 PM	3:00 PM	4:00 PM	5:00 PM	6:00 PM	7:00 PM	8:00 PM	9:00 PM	10:00 PM	11:00 PM
Monday	🟪🟥	🟪🟥			🟪	🟪						
Tuesday	▨	▨										
Wednesday	▨	▨	▨									
Thursday												
Friday												
Saturday												
Sunday												

Figure 9: Collated Responses for Poll

To facilitate the date/time slot voting process, TiddlyCalendar provides an integrated week view which displays the current appointments of the participant, making it easy for the participant to identify any conflicting date/time slots and decide his/her availability (Figure 10). The use of colour schemes (purple depicts poll options while other colours depict appointments of different categories) aids in quick identification of the date/time slots.



POLL DETAILS

Summary: Test Appointment Poll
Categories: Meeting
Location: SOC
Details: This is a test poll.

	Option 1 29/10/2007 12:00 29/10/2007 14:00	Option 2 29/10/2007 16:00 29/10/2007 18:00	Option 3 30/10/2007 10:00 30/10/2007 12:00	Option 4 31/10/2007 10:00 31/10/2007 12:00
Steven		OK	OK	OK
Janet	OK	OK		
Peter		OK	OK	
Total:	1	3	2	1
Your Selection:	☐	☐	☐	☐

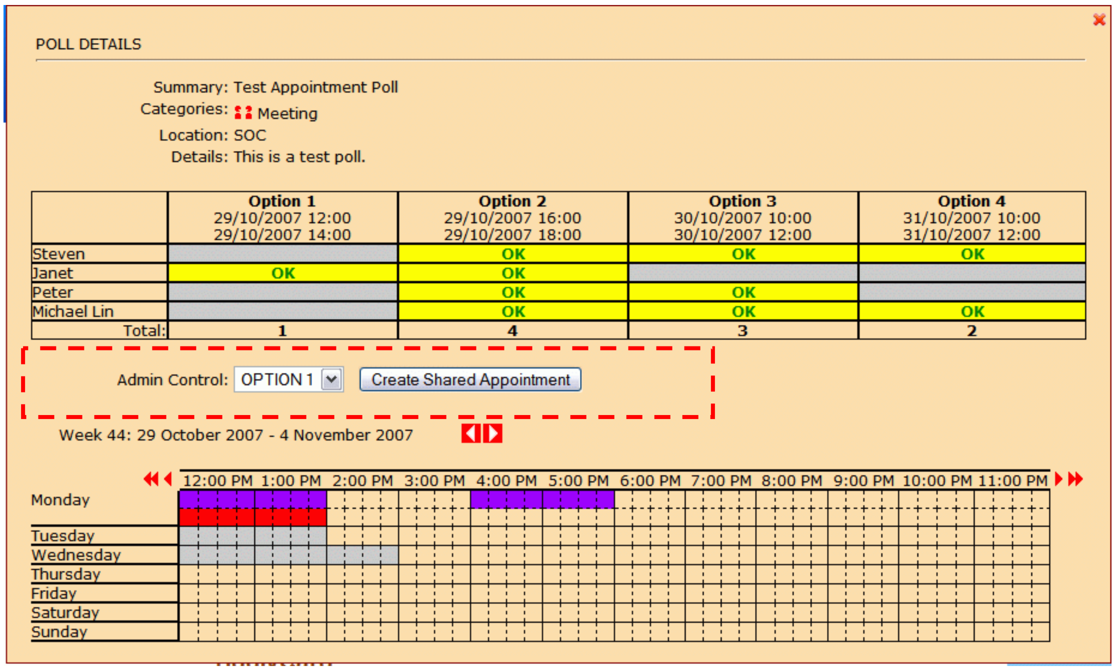
Confirm

Week 44: 29 October 2007 - 4 November 2007

	12:00 PM	1:00 PM	2:00 PM	3:00 PM	4:00 PM	5:00 PM	6:00 PM	7:00 PM	8:00 PM	9:00 PM	10:00 PM	11:00 PM
Monday	🟪🟥	🟪🟥			🟪	🟪						
Tuesday	▨	▨										
Wednesday	▨	▨	▨									
Thursday												
Friday												
Saturday												
Sunday												

Figure 10: Integrated Week View

Such polling system provides the appointment coordinator with updated information on users' selection and thereby assists the coordinator in making a final decision on the appointment date/time quickly and easily. Once the coordinator is satisfied, he/she may finalise the poll using the Administrative Control (Figure 11). Upon confirmation, a new public event is created and sent to all participants automatically, reducing the need to notify all the participants and the number of steps to complete the scheduling process.



POLL DETAILS

Summary: Test Appointment Poll
Categories: Meeting
Location: SOC
Details: This is a test poll.

	Option 1 29/10/2007 12:00 29/10/2007 14:00	Option 2 29/10/2007 16:00 29/10/2007 18:00	Option 3 30/10/2007 10:00 30/10/2007 12:00	Option 4 31/10/2007 10:00 31/10/2007 12:00
Steven		OK	OK	OK
Janet	OK	OK		
Peter		OK	OK	
Michael Lin		OK	OK	OK
Total:	1	4	3	2

Admin Control: OPTION 1 Create Shared Appointment

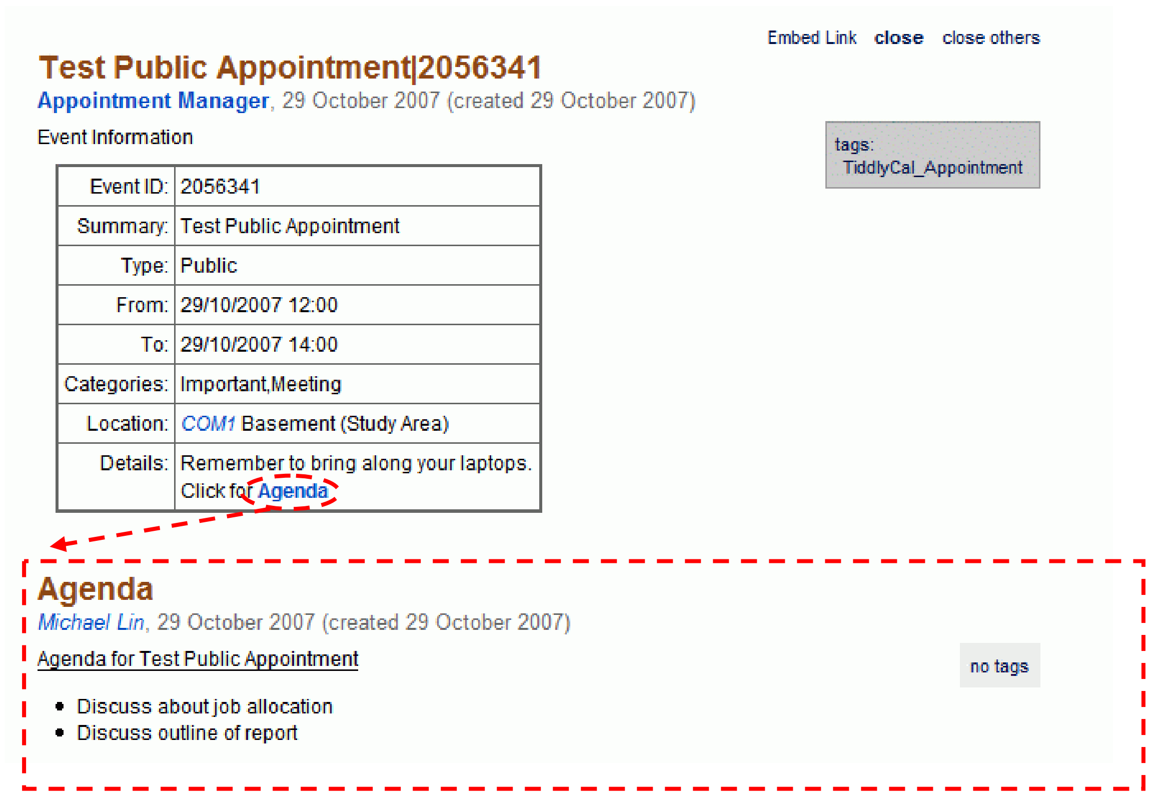
Week 44: 29 October 2007 - 4 November 2007

Figure 11: Administrative Control

3.3 Integration with TiddlyCard Components

3.3.1 Integration with Wiki Component

TiddlyCalendar aims to provide an integrated method of appointment scheduling by eliminating the need to revert to multiple applications. Through the public appointment sharing feature, TiddlyCalendar has removed the need for a third party application such as an email application to notify the participants of new or updated events. To take this one step further, TiddlyCalendar integrates with the Wiki component of TiddlyCard which is used to create, store and transfer content such as agenda and meeting minutes by maintaining references to the content tiddler in the calendar. This is possible through the creation of appointment tiddlers (Figure 12) which "wikifies" the appointment content.


Embed Link close close others

Test Public Appointment|2056341

Appointment Manager, 29 October 2007 (created 29 October 2007)

tags: TiddlyCal_Appointment

Event Information

Event ID:	2056341
Summary:	Test Public Appointment
Type:	Public
From:	29/10/2007 12:00
To:	29/10/2007 14:00
Categories:	Important,Meeting
Location:	COM1 Basement (Study Area)
Details:	Remember to bring along your laptops. Click for Agenda

Agenda

Michael Lin, 29 October 2007 (created 29 October 2007)

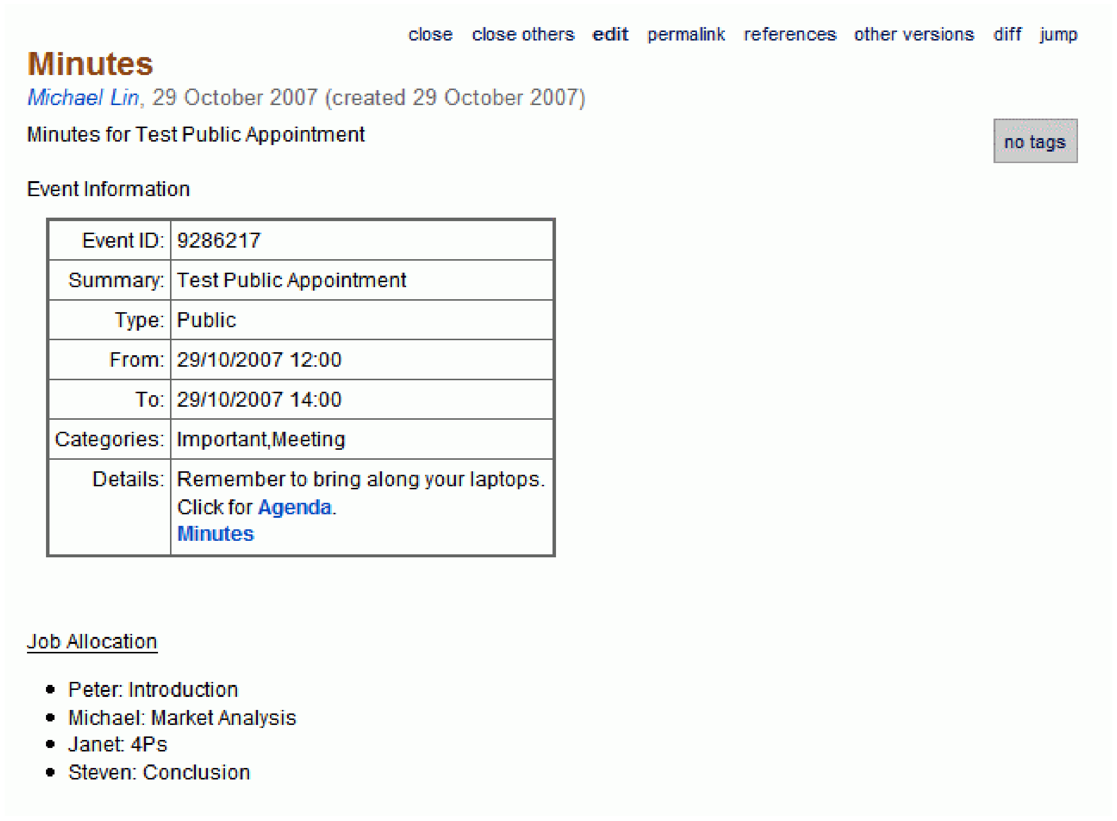
no tags

Agenda for Test Public Appointment

- Discuss about job allocation
- Discuss outline of report

Figure 12: Appointment Tiddler Linking to Agenda Tiddler

Through the use of the Event Macro, it is also possible to include the appointment information in a tiddler through the use of Embed Link (Figure 13)


close close others edit permalink references other versions diff jump

Minutes

Michael Lin, 29 October 2007 (created 29 October 2007)

no tags

Minutes for Test Public Appointment

Event Information

Event ID:	9286217
Summary:	Test Public Appointment
Type:	Public
From:	29/10/2007 12:00
To:	29/10/2007 14:00
Categories:	Important,Meeting
Details:	Remember to bring along your laptops. Click for Agenda. Minutes

Job Allocation

- Peter: Introduction
- Michael: Market Analysis
- Janet: 4Ps
- Steven: Conclusion

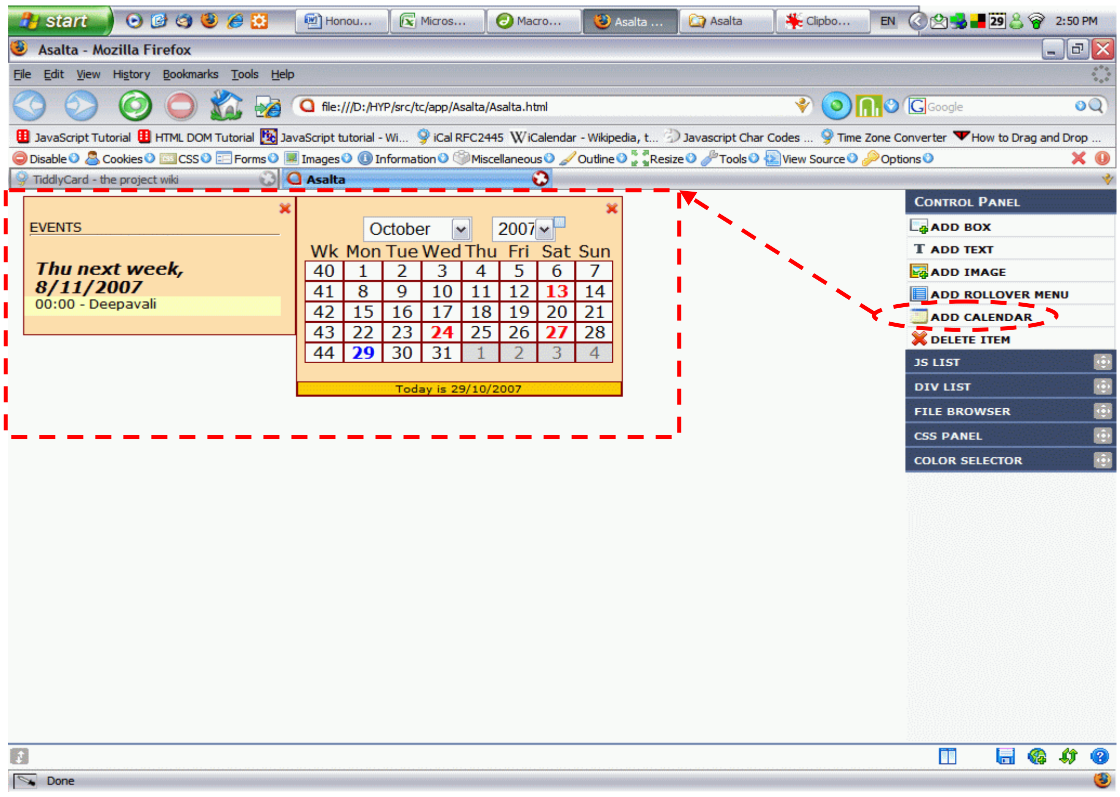
Figure 13: Embedding Appointment Information in Minutes Tiddler

The benefits of integration of TiddlyCalendar with the Wiki component are summarised below:

a. Eliminate third party applications such as word processing applications for creating appointment-related content such as Agenda and Minutes
b. Provide ability to embed appointment information in content tiddlers (e.g. Minutes) for easy referencing
c. Provide ability to link appointment information and appointment related content tiddlers (e.g. Agenda)

3.3.2 Integration with Asalta Component

TiddlyCalendar provides an API Library which enables Asalta to provide calendar feature to the webpage that Asalta is building. This enables a web designer or developer to deploy TiddlyCalendar quickly within the application he/she is building simply by clicking the Add Calendar button (Figure 14). This also allows Asalta to perform changes on TiddlyCalendar's properties e.g. resizing, changing of colour schemes, etc, enabling developers to customise the TiddlyCalendar's appearance while preserving its functionalities.



EVENTS

Thu next week, 8/11/2007
00:00 - Deepavali

October 2007

Wk	Mon	Tue	Wed	Thu	Fri	Sat	Sun
40	1	2	3	4	5	6	7
41	8	9	10	11	12	13	14
42	15	16	17	18	19	20	21
43	22	23	24	25	26	27	28
44	29	30	31	1	2	3	4

Today is 29/10/2007

Figure 14: TiddlyCalendar Integration with API Library

3.3.3 Integration with TiddlyRSS Component

TiddlyCalendar integrates with TiddlyRSS by making use of TiddlyRSS's API to generate RSS feeds when the calendar is changed. This RSS file may then be uploaded to a webserver which can be subscribed to by another user.

While other RSS readers display the feed as normal updates, TiddlyRSS can detect special <event> </event> tags which contain appointment details and call the TiddlyCalendar API to perform updates to the calendar. This allows user to be freed from having to manually check for changes to a calendar of interest as the TiddlyRSS retrieves updates and imports new/changed appointments automatically.

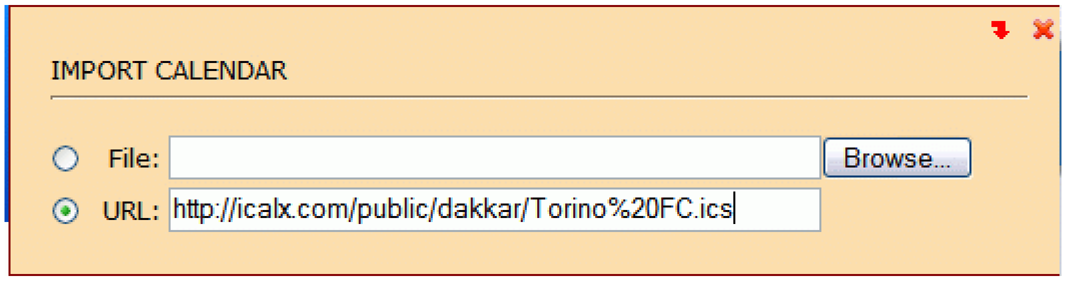
3.4 Multiple Import Mechanisms

TiddlyCalendar supports 2 primary different methods of importing appointments.

a. Importing appointments through iCal file format (see Section 3.4.1)
b. Importing appointment though appointment tiddlers (see Section 3.4.2)

3.4.1 Importing Appointments through iCal

TiddlyCalendar supports importing appointments in iCal format in both local and remote iCal files. iCal files can be downloaded to the local computer system before being added to the calendar or loaded directly using the HTTP protocol (Figure 15).


IMPORT CALENDAR

File:

[✓ URL:] http://icalx.com/public/dakkar/Torino%20FC.ics

Figure 15: Importing through iCal File

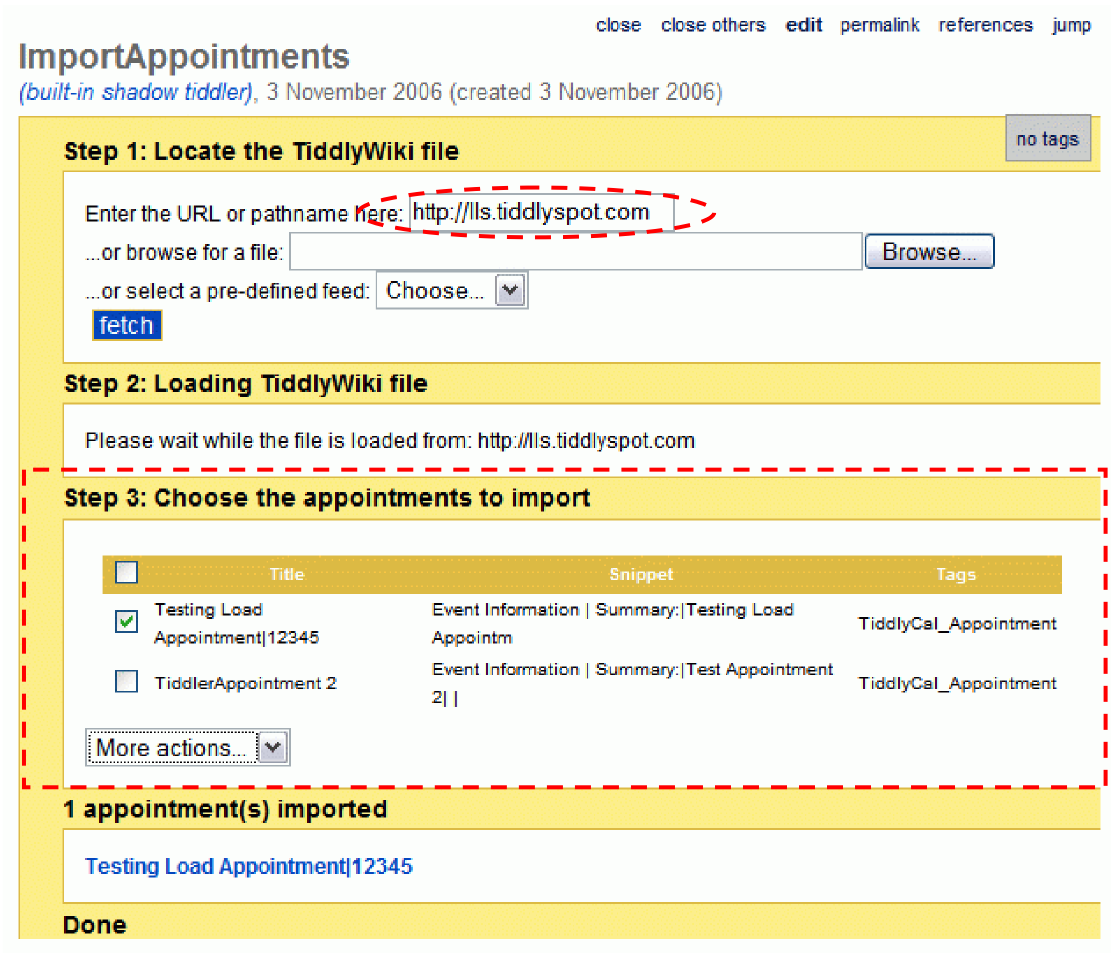
3.4.2 Importing Appointment through Appointment Tiddlers

This import method allows appointment to be imported via tiddlers. It works based on tiddlers that are tagged with a special tag called "TiddlyCal_Appointment". It searches local or remote tiddlers marked with this special tag and allows users to select specific appointments to import into TiddlyCalendar (Figure 16).

This import method can be performed in 2 easy steps:

a. Specific the URL or local file path

b. Select the appointments to import from the list of appointment tiddlers



ImportAppointments

(built-in shadow tiddler), 3 November 2006 (created 3 November 2006)

Step 1: Locate the TiddlyWiki file

Enter the URL or pathname here: http://lls.tiddlyspot.com

...or browse for a file:

...or select a pre-defined feed: Choose...

Step 2: Loading TiddlyWiki file

Please wait while the file is loaded from: http://lls.tiddlyspot.com

Step 3: Choose the appointments to import

☐	Title	Snippet	Tags
✓	Testing Load Appointment|12345	Event Information | Summary:|Testing Load Appointm	TiddlyCal_Appointment
☐	TiddlerAppointment 2	Event Information | Summary:|Test Appointment 2| |	TiddlyCal_Appointment

1 appointment(s) imported

Testing Load Appointment|12345

Done

Figure 16: Importing through Appointment Tiddlers

4. Design and Development

4.1 Language, Technologies and Formats

4.1.1 Javascript

The use of Javascript dated back to the early days of the internet when it is mainly used as a client-based tool to perform tasks such as data validation in order to reduce web traffic to-and-fro the server. As bandwidth increases and connection speeds get faster and faster, many companies switched back to server-side processing which cuts down the use of Javascript heavily. However, the language is regaining its popularity with the increasing use of the Document Object Model (DOM), which enables browsers to create content dynamically on the client without the need to request such information from the server.

Javascript is chosen as the language for development for TiddlyCalendar for the following reasons:

a. Server and connectivity independence

   Javascript eliminates the need for server and internet connectivity as it is able to operate ceaselessly in a browser. This enables users to use TiddlyCalendar continuously, even if connectivity is broken midway.

b. Lightweight

   Javascript files occupy a small footprint (typically few hundred kilobytes to less than one megabyte) compared to other languages. This makes it easy for users to port TiddlyCalendar to other machines/systems together with its data files.

c. Platform independence

   All modern browsers provide native support for Javascript. This enables Javascript code to be executed with the need for installation of any runtime environments or Software Development Kit (SDK). Javascript platform independence allows it to be used on any device (PCs, laptops, PDAs or mobile phones) running on any operating system (Windows, Linux, Symbian).

d. OOP in Javascript

   Although Javascript is not completely object-oriented, it provides a high level of support by being object-based. This allows TiddlyCalendar to create reusable objects and classes which can be shared among different classes and even applications. An example is the Appt class:

```
thisClass.Appt = function(){
      this.apptId = 0;
      this.apptUID = "";
      this.apptEvent = "";
      this.apptLoc = "";
      this.apptDateCreated = new Date();
      this.apptDateModified = new Date();
      this.apptStartDateTime = new Date();
      this.apptEndDateTime = new Date();
      this.apptAllDay = false;
      this.apptDetail = "";
      this.apptCategory = "";
      this.apptType = "";
      return this;
}
```

4.1.2 XMLHttpRequest

XMLHttpRequest is an important Ajax web development technique that provides an API that can be used by Javascript to transfer XML and other text data to and from a webserver using HTTP by establishing an independent and asynchronous communication channel between the client and the server (XMLHttpRequest, 2007).

TiddlyCalendar’s import mechanisms use XMLHttpRequest to retrieve the contents of other TiddlyCard/TiddlyWiki appointment tiddlers as well as appointments that are stored in iCal files that are hosted on other webservers. This enables clients to import appointments by specifying the URL of the file containing the appointments instead of having to download the file before importing the appointments (Figure 17).

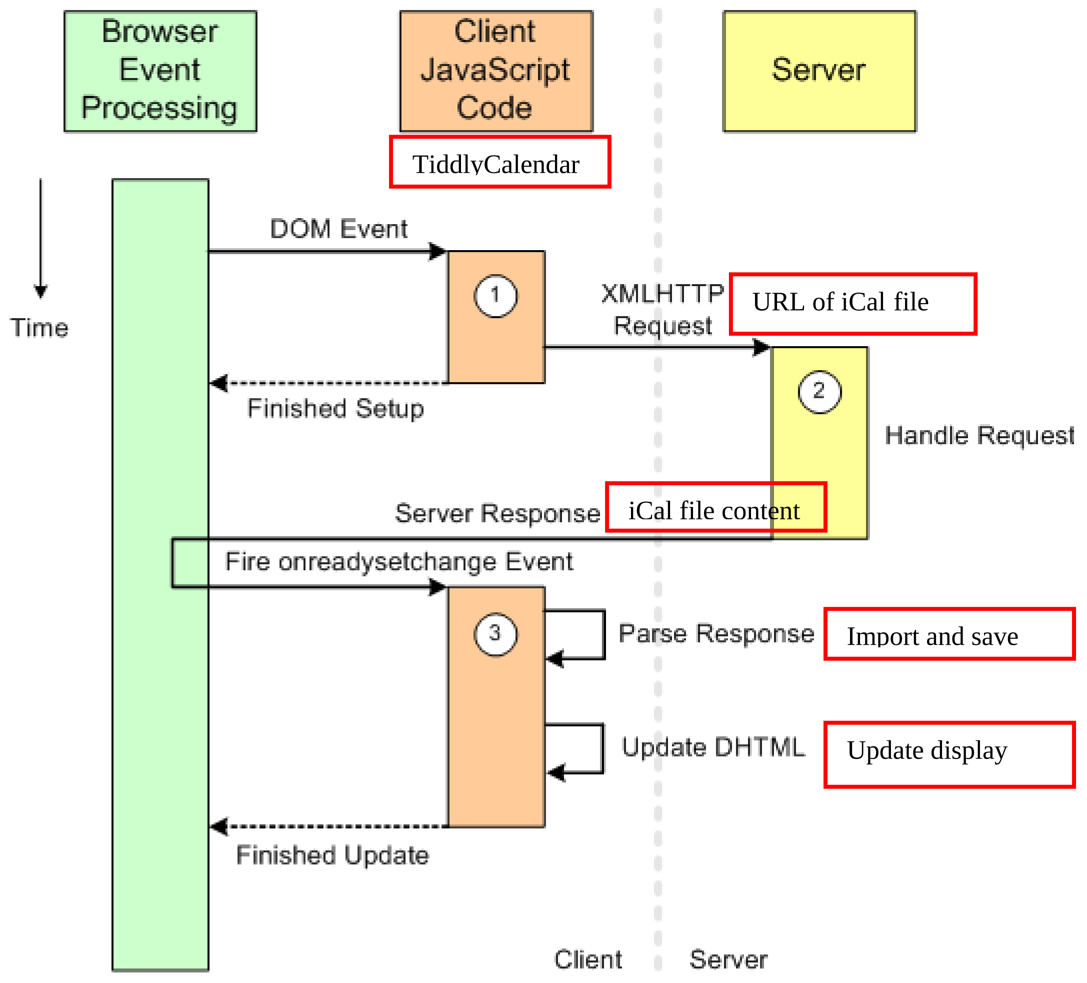


Figure 17: Importing Appointments from Remote Files using XMLHttpRequest (source: http://www.modelworks.com/ajax.html)

4.1.3 Comet

TiddlyCalendar makes use of the Comet architecture which enables the server to send data to clients asynchronously without the client explicitly requesting for it. Typically, Comet applications use long-lived HTTP connections between the client and server, providing an opened channel for clients and servers to send/receive messages. When client A sends a message, the server relays the message to all clients that are online and connected to the server (Figure 18).

The Comet architecture used by TiddlyCalendar is a modified version where a persistent store is created to store messages sent by clients. This allows clients to retrieve history of messages that are sent to the server while they are offline and allows them to perform synchronisation (Figure 19).

The public event and polling features of TiddlyCalendar make use of Comet to push new/updated events/polls to the clients so that clients can receive updates and perform synchronisation when connectivity is available. However, the normal operations of TiddlyCalendar are not hindered in any way when connectivity is not available.

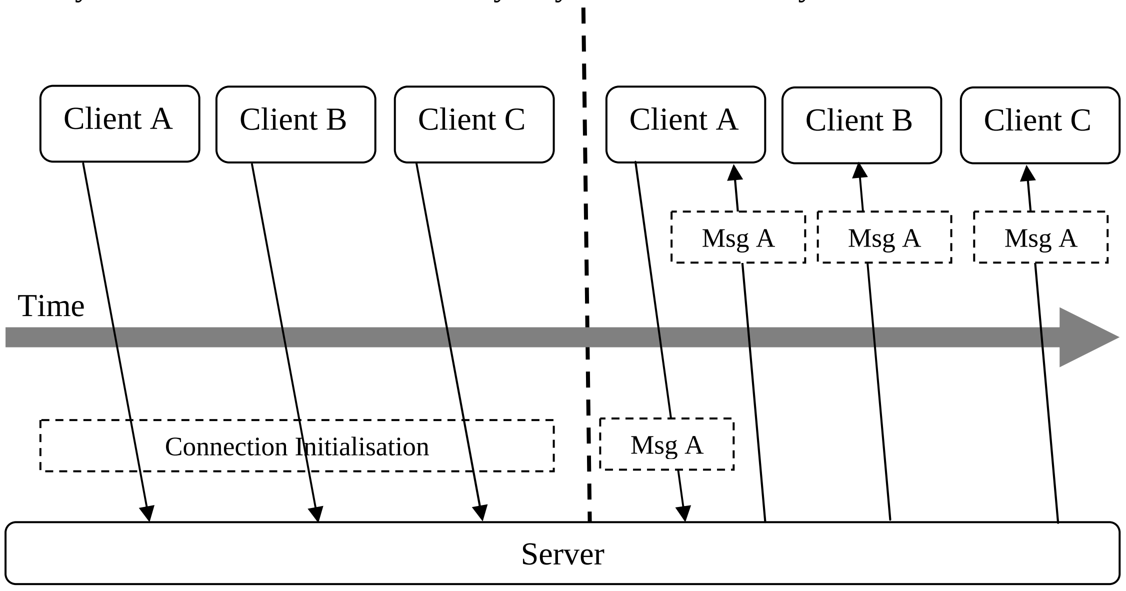


Figure 18: Comet Architecture

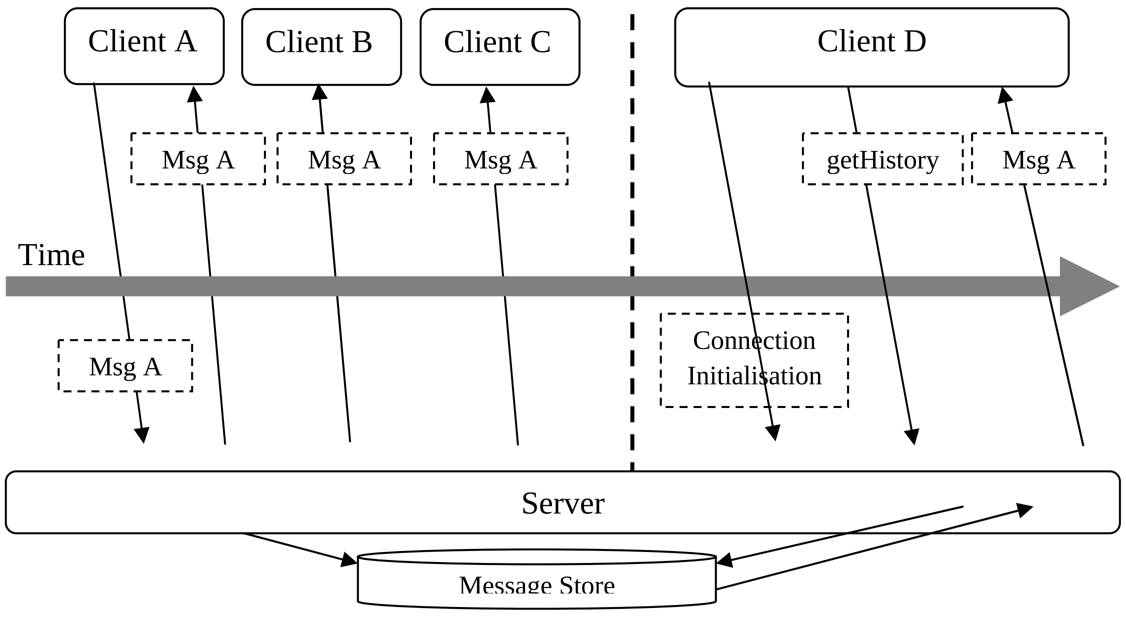


Figure 19: Modified Comet with Persistent Store

4.1.4 JavaScript Object Notation (JSON)

JSON has been chosen as the file format used by TiddlyCalendar's datastores. Based on subset of Javascript, JSON provides a light-weight data-interchange format that is easy for human to read/write and machines to parse and generate (Introducing JSON, n.d).

JSON makes it easy for Javascript to access the data and objects stored in a JSON file as the internal representations for data structures like strings, arrays, and objects are exactly the same characters (Figure 20). Objects can be restored back to their original state (including variable values) by simply using the eval() function. This eliminates the complexity of storing the data in other formats e.g. XML which introduces overheads and requires parsing before the objects can be retrieved and reinstated in its original state.

```
[{"apptId": "8553265", "apptUID": "8553265", "apptEvent": "Testing",
"apptLoc":   "Home",   "apptDateCreated":   new   Date(1193889543665),
"apptDateModified": new Date(1193889543665), "apptStartDateTime": new
Date(1193760003665),   "apptEndDateTime":   new   Date(1193767203665),
"apptAllDay":    false,    "apptDetail":    "Some    test    details",
"apptCategory": "1", "apptType": 0}]
```

Figure 20: JSON Representation of Appointment Datastore

4.1.5 Calendar Algorithms

The month display window uses a unique algorithm (Cross browser javascipt calendar, n.d.) to display the days in the calendar month. Extracts of the main steps in the code are illustrated below:

```
//1. Declare array containing number of days in month
var calDaysNo = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31];

//2. Leap year calculation and set number of days in month to 29 if
leap year
if (selectedMonth == 1) { //Testing for February
      if ((selectedYear % 4 == 0 && selectedYear % 100 != 0) ||
selectedYear % 400 == 0){
            monthDayNo = 29;
      }
}

//3. Day Number Filling for selected month/year
for(var k=0;k<=6;k++){  //j = week rows, k = day cols
      if(day<=monthDayNo && (k>=((firstDay-1)%7) || j>0)){
            cell.innerHTML = day;
            ++day;

//4. Day Number Filling for previous month
...
}else{
      if(k<((firstDay-1)%7) && j==0){     //must be first row
            cell.innerHTML = pastDate.getDate();
```

```
//5. Day Number Filling for next month
...
if(day>monthDayNo){
      cell.innerHTML = nextMonthDate.getDate();
...
```

The week number of the calendar window and week view window is calculated based on the ISO 8601 Week standard and uses a special algorithm derived from Reingold and Dershowitz (2001).

```
thisClass._computeWeek = function(year,month,day){
      day = day/1;
      year = year/1;
      month = month/1+1; //use 1-12
      var a = Math.floor((14-(month))/12);
      var y = year+4800-a;
      var m = (month)+(12*a)-3;
      var jd = day + Math.floor(((153*m)+2)/5) +
      (365*y) + Math.floor(y/4) - Math.floor(y/100) +
      Math.floor(y/400) - 32045;      // (gregorian calendar)

      var d4 = (jd+31741-(jd%7))%146097%36524%1461;
      var L = Math.floor(d4/1460);
      var d1 = ((d4-L)%365)+L;
      var NumberOfWeek = Math.floor(d1/7) + 1;
      return NumberOfWeek;
}
```

4.1.6 iCalendar

TiddlyCalendar supports the use of iCal file format that is widely popular in calendar sharing. TiddlyCalendar provides an export mechanism that allows users to export their TiddlyCalendar appointments to iCal format for compatibility with other calendar applications such as Microsoft Outlook and Rainlendar. The reverse is possible as well where appointments created in other calendar applications can be imported via the iCal format for cross-application interoperability. TiddlyCalendar achieves this through conforming to RFC 2445 which provides the iCalendar specification (Dawson and Stenerson, 1998).

4.2 Software Development Process

Although the majority of the requirements specification for the TiddlyCalendar is drafted in the early stage of the project period, TiddlyCalendar followed the iterative and incremental development process (Priestley, 2003) instead of the waterfall model as development and delivery is broken down into small increments (Figure 21).

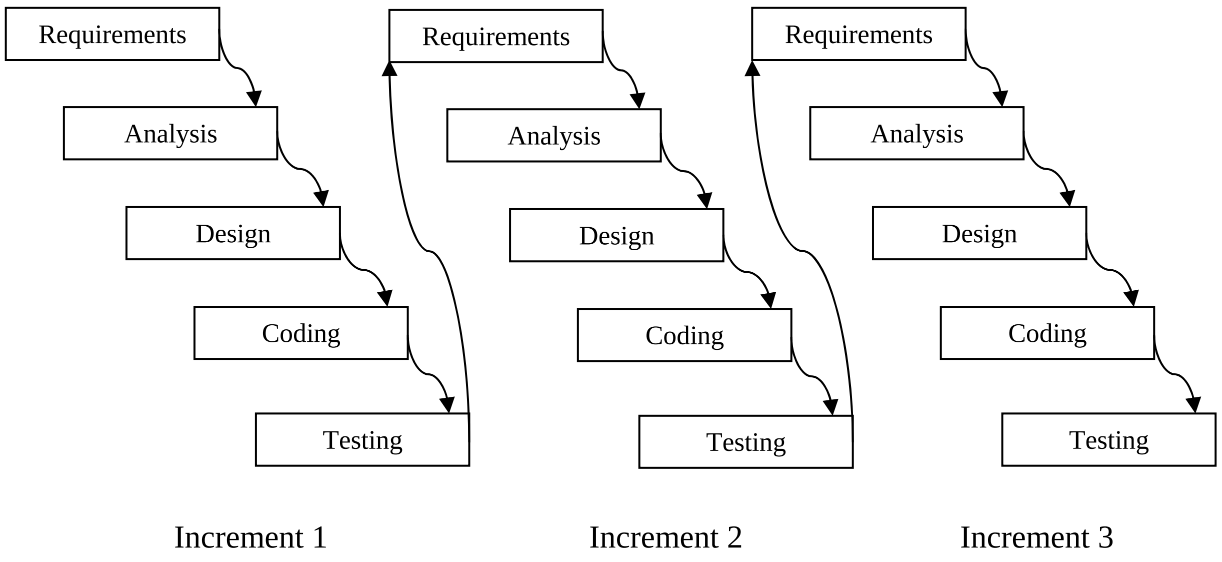


Figure 21: Iterative and Incremental Development Process

This draft specification provided a baseline to conceptualise the end result of the development, providing input from which future feature improvements can build upon on during later stages of development.

Core functionalities such as display of the calendar, appointment manager can be produced early in the project life and released to potential users for testing and feedback gathering. This produces working code that can be refined continuously throughout the project, leading to lower risk of failure with abundant testing throughout the entire project period. Although it may seem that the work is repetitive and may cause delays in development progress, the contrary is true instead as slack time between development phases is reduced by performing bug fixing.

The iterative and incremental development process allows prototypes to be build to test for feasibility and to gather feedback from the TiddlyCard Developer Community. Suggestions can be noted and slated for incremental development later.

The development of each incremental phase in TiddlyCalendar is shown below:

Increment 1:	Develop calendar display which shows the calendar for selected month/year
Increment 2:	Develop Appointment Manager to provide Add/Update/Delete/Search functions
Increment 3:	Integrate calendar display with Appointment Manager. Develop new event list window in calendar display to show upcoming appointments
Increment 4:	Develop Alarm Manager to provide alarm and reminder for appointments
Increment 5:	Develop Public Appointment Sharing feature
Increment 6:	Develop Appointment Polling feature

* Testing and bug fixes occur throughout the entire development process, mainly between phases

4.3 Design Pattern

TiddlyCalendar employs the Model-View-Controller (MVC) (Dass, 2006) design pattern in its development (Figure 22).

Interface (View)	Control (Controller)	Data (Model)
Calendar Display Manager • Month view • Week view • Event List view • Detail view • Summary view	Appointment Manager • Add/edit/delete appt • Search appt • Import/export • Polling	Appointment Poll
	Alarm Manager • Add/edit/delete • Search • Time tracking • Alarm notification	Alarm

Figure 22: TiddlyCalendar MVC Architecture

The model stores all data about the object and corresponds to the Datastores. These include:

a. Appt
b. Poll
c. Alarm

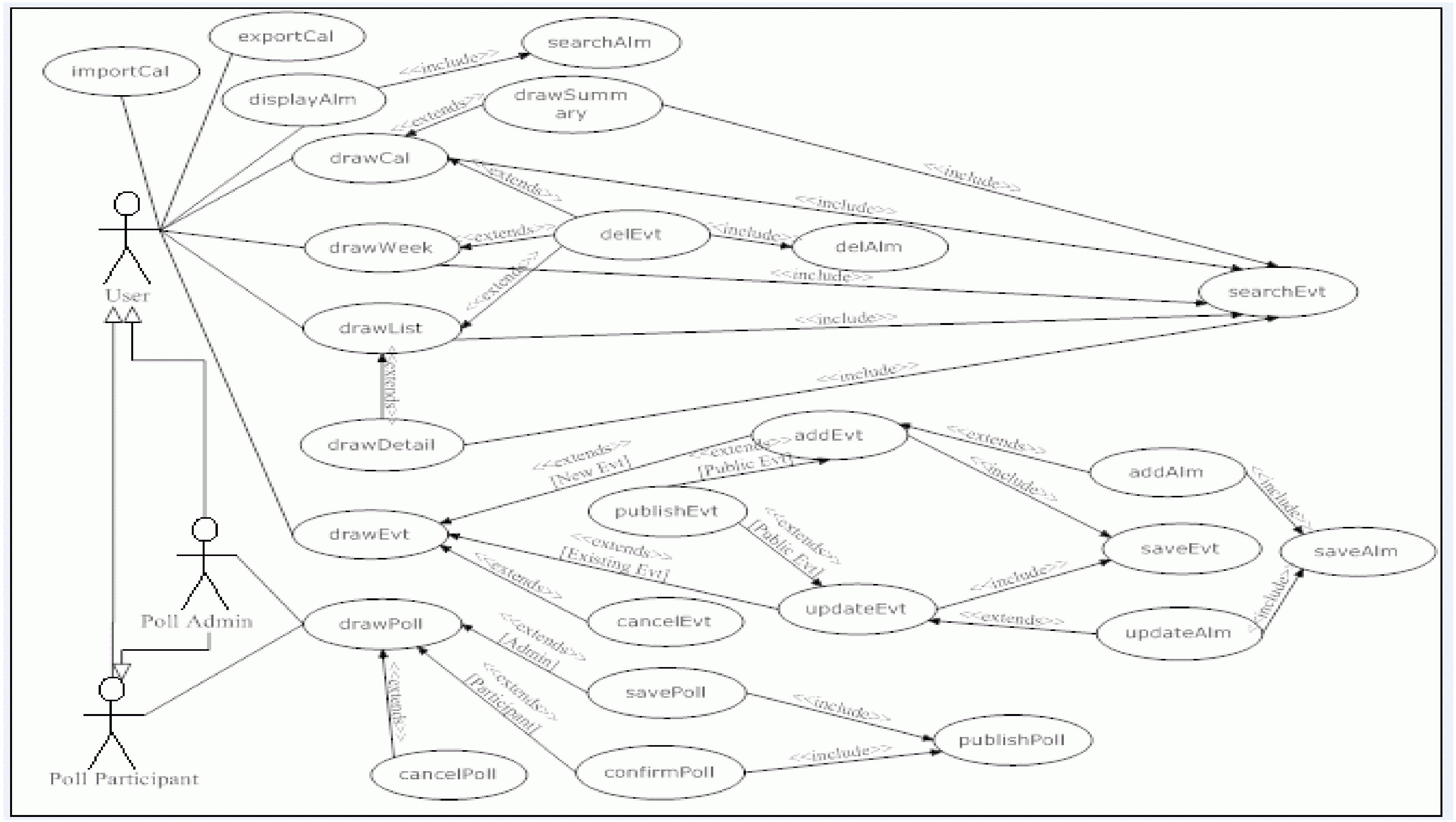
The view manages the graphical display of the application. The calendar display manager handles this aspect, displaying the calendar month (highlighting dates with appointments), the event list window (displaying upcoming appointments), week view and details window. The calendar display manger is always notified when changes to the models occur.

The appointment manager and alarm manager are the controllers for TiddlyCalendar. They are responsible for responding to user requests such as creating new appointments/alarms/polls and requests from the view (e.g. search appointments list for appointments in display range). Changes to the models are performed by the controllers.

The MVC design pattern provided several benefits to the development of TiddlyCalendar:

a. Increased modularity - Each individual component can be designed and developed independently of each other. Changes to component have a less likely impact on other components e.g. model can be changed easily without major changes to the calendar display manager (view)
b. Improved flexibility – It is easy to change the UI of TiddlyCalendar by changing the view e.g. Asalta is able to perform changes to the display of TiddlyCalendar (e.g. colour scheme) without affecting the operations of TiddlyCalendar
c. Increased testability – Individual components can be tested separately from the UI. For example, tests can be performed on the appointment manager to check if search results (from models) are correctly returned before integrating with the calendar display manager

4.4 UML Use Case Diagram



5. Lessons Learnt

5.1 Collaboration

Although the project is individual-based, being a part of a bigger development community presented the opportunity to collaborate with others. Frequent discussions and meetings about the direction of TiddlyCard are held when each developer contributed actively, often working out expectations and resolving conflicts before finalising any decisions which may affect each other's work. An example will be the change of the file IO library from synchronous read/write to asynchronous read/write, which broke existing TiddlyCalendar code before it was changed to include callbacks.

Cooperation between developers also proved beneficial to the entire project. An example will be the Comet architecture that was modified to include a persistent store. Though initially discussed as a possible tool for TiddlyCalendar, this modification eventually allowed other applications such as the Messenger to benefit as well.

5.2 Standards and Formats

TiddlyCalendar follows the ISO 8601 Week and iCal standards. By basing development on such standards, TiddlyCalendar has a strong foundation to base its work on, where the specifications provided guidelines for development and aid in ensuring compatibility with existing calendar applications (which also makes use of the same standards). Apart from making use of existing standards, TiddlyCalendar also defined its own format to allow tiddlers to be used as a channel for the import mechanism between TiddlyCalendar applications.

The week view of TiddlyCalendar was initially designed to have a horizontal layout (Figure 23). However, this user interface soon encountered resistance as it conflicted with the vertical layout (Figure 24) format which is ingrained in calendar usage.

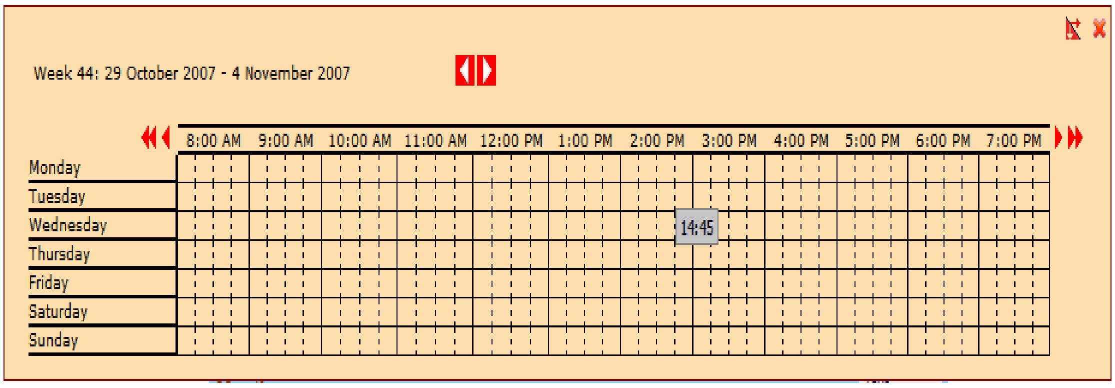


Figure 23: Horizontal Week View

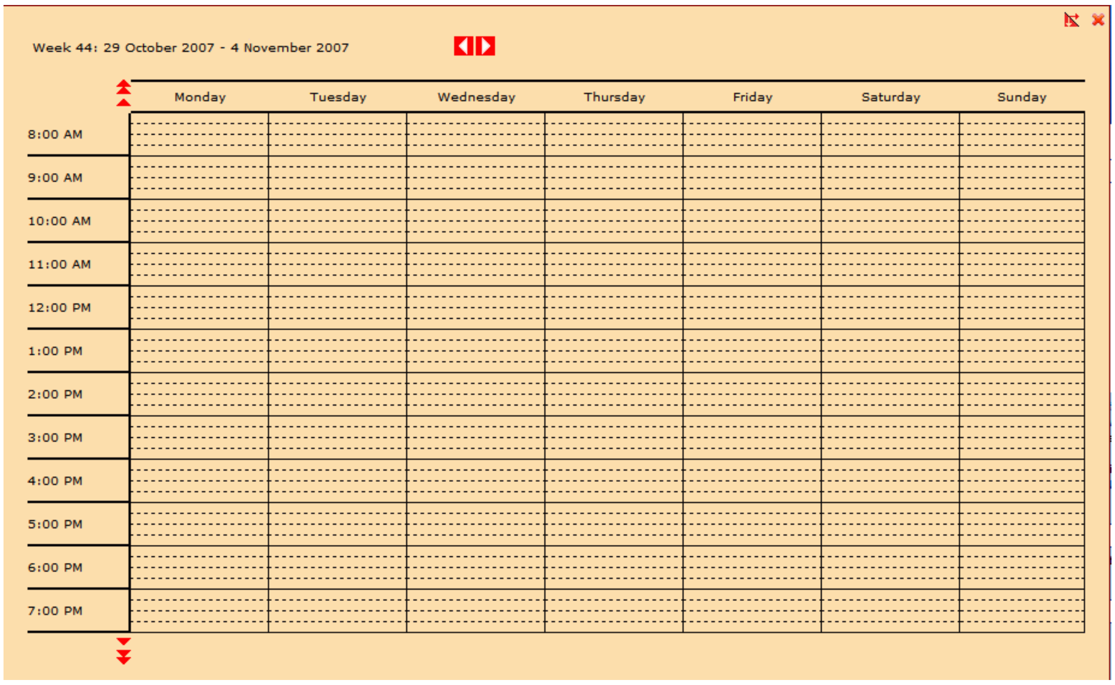


Figure 24: Vertical Week View

Although the horizontal layout provided a compact view which is able to provide a clearer landscape of the user's weekly events, users' perspectives and acceptance level of the new introduction should not be over-estimated (e.g. Qwerty and Dvorak keyboard layout). This led to the consideration and eventual adoption of the idea to allow users to switch between the two views, which provided an ease of transition for the users to the new layout while preserving the vertical layout for change-resistant users.

6. Recommendations for Future Works

6.1 User Interface

6.1.1 Single Window Design

TiddlyCalendar currently contains 7 different windows to perform various tasks. These are:

a. Calendar month display window
b. Event list window
c. Week view window
d. Add/Update event window
e. Poll window
f. Import window
g. Export window

This multiple window interface clutters the workspace and creates an untidy and messy interface. In addition, the user has to close the windows one by one in order to fully exit TiddlyCalendar.

It is recommended to design and implement a single window interface where all the functions are housed in one single window for easy operability. This single window design will allow users to access TiddlyCalendar functions from one location, while maintaining the tidiness of the workspace.

6.1.2 Drag-and-drop Capability

TiddlyCalendar supports moving of windows through the use of drag-and-drop function. Different windows can be arranged in the workspace by simply dragging and dropping the window to the destination location, while the window maintains visibility to allow user to see the exact location after movement.

This drag-and-drop capability can be extended to appointments in the week view to allow changing of date/time of the appointment. This provides added convenience to the user by eliminating the need to change the date/time through the add/update event window.

6.2 Feature Improvements

6.2.1 Appointment Polling

The innovative polling system allows the poll administrator to specify multiple date/time slots for the appointment. However, the decision making process of the selection of the slots is still based on one single individual i.e. poll administrator. This system can be improved by introducing the "Call for Options" stage, which allows the administrator to send out a request for free time slots from each participant, thereby creating date/time slots that are more likely to be accepted.

Poll options are currently presented in one single colour (purple) in the week view integrated in the poll window. It is suggested that different options adopt different colours to allow the user to identify the options quickly without the need to perform a "mouse-over" the option to view the option details.

6.2.2 Public Appointment Sharing

When a public appointment is updated, the entire appointment object is changed and all users sharing the event are updated. This scenario might not be desired if the user wishes to include individual notes, preferences e.g. alarm or details. This can be improved by allowing the user to state if TiddlyCalendar should notify all users who are sharing this appointment. In this way, users can perform actions such as adding alarms (at different timings) based on individual preferences without affecting others.

In order to facilitate this behaviour, it is also suggested that changes to shared appointments are based on attributes rather than entire objects. Instead of updating the entire appointment object when changes are made, TiddlyCalendar should allow changing of individual fields which will be updated without affecting other fields e.g. changing the time of the public appointment will only send an update to change the time for other users, while preserving their individual notes such as personal additional details/alarms.

6.2.3 Export Selection

The current mechanism exports all appointments in the user's TiddlyCalendar. This action might not be desirable as certain appointments might be private which the user does not wish to disclose.

A possible solution is to allow the user to specify the appointments to be exported. This can be aided by the inclusion of filters, which assists the user in selecting the appointments e.g. all public appointments only, all appointments of Meeting category, etc.

Sorting mechanisms such as Summary alphabetical sort, Date/Time sort, Category sort, etc will also support the export mechanism, allowing users to select the appointments to be exported easily.

6.3 Feature Additions

6.3.1 Comet Queue

Currently, all public appointments and polls are only communicated and shared if they are created at a time when the user is online. Due to the Comet architecture, the messages are published to the server immediately after creating the appointment/poll. This meant that when an offline user creates a public appointment/poll, the information is “lost” as it cannot be sent to the server, disallowing other users to access the appointment information.

It is suggested that a offline queue be implemented which will store the messages to be sent to the Comet server. A network monitor can then be installed to track network connectivity. If a connection becomes available, the messages in the queue can then be sent to the server.

6.3.2 Free/Busy Time Notification

This feature can be added to allow users to notify other users their free/busy time periods explicitly. This helps meeting coordinators to plan date/time slots more effectively, reducing the number of interactions required to find suitable time slots for appointments.

6.3.3 Others

The following features may be considered for implementation as well.

a. Multiple Calendars
b. Day View
c. Recurring Appointments
d. Category Rules e.g. yearly recurrence for Anniversary and Birthday events

7. Comparison and Conclusion

TiddlyCalendar has written a new chapter in the history of Javascript calendars. Unlike traditional Javascript calendar which only aims to display calendar month information (much like physical calendar) and allow users to select dates i.e. date-pickers in forms, TiddlyCalendar provides PIM calendar features that is comparable to state-of-the-art desktop and server applications such as Microsoft Outlook and Google Calendar which allows user to manage their schedules in an integrated environment through the TiddlyCard platform, allowing seamless integration between content and schedule.

Although calendar is in a well-developed and matured field, TiddlyCalendar has presented innovative ideas in sharing of appointments and scheduling of public appointments with the Public Appointment Sharing and Appointment Polling features, breaking the stagnant state in PIM calendar applications.

TiddlyCalendar provides a scheduling tool that is integrated with TiddlyCard to provide seamless integration between content and schedule.

Its innovative appointment polling/sharing features make it easy for users to arrange appointments/meetings and collaborate on tasks using the same platform. Users can share appointment information with each other easily using a large variety of mechanisms such as Comet, iCal files, Appointment tiddlers and even RSS feeds.

TiddlyCalendar has been integrated with other TiddlyCard applications such as Asalta and TiddlyRSS easily through the use of Libraries and APIs. This use of libraries and APIs allows future integration with other applications (e.g. embedding appointment information in email application), which can aid in collaborative efforts among multiple users, allowing greater efficiency and effectiveness in group work.

Although TiddlyCalendar has not fulfilled all the features implemented in a matured calendar application such as Microsoft Outlook due to the time span of the HYP period, improvements to existing TiddlyCalendar has been suggested in Section 6 to realise its full potential in future developments, bringing calendar applications to new heights.

References

Text References

Adobe Labs. (2006). *Adobe AIR*. Retrieved October 25, 2007, from Adobe Labs: http://labs.adobe.com/technologies/air/

Comet (programming). (2007, October 25). Retrieved October 29, 2007, from Wikipedia: http://en.wikipedia.org/wiki/Comet_(programming)

Dass, K. (2006). *MVC Acrhitecture*. Retrieved November 2, 2007, from IndiaWebDevelopers.com: http://www.indiawebdevelopers.com/technology/java/mvcarchitecture.asp

Dawson, F., & Stenerson, D. (1998, November 1). Internet Calendaring and Scheduling Core Object Specification (iCalendar).

Dusseault, L. (2005, March 7). *Welcome to CalDav Resources*. Retrieved October 26, 2007, from CalDav: http://ietf.osafoundation.org/caldav/index.html

HyperCard. (12 October, 2007). Retrieved 25 October, 2007, from Wikipedia: http://en.wikipedia.org/wiki/HyperCard

Introducing JSON. (n.d.). Retrieved November 1, 2007, from JSON: www.json.org

Inturico Engineering. (2006). *Doodle*. Retrieved October 1, 2007, from Doodle: http://www.doodle.ch

Nen, L. V. (Ed.). (2000, December 15). Data elements and interchange formats – Information interchange –. *ISO8601:2000(E)* (2). International Organization for Standardization.

Priestley, M. (2003). Software Development Processes. In M. Priestley, *Practical Object-Oriented Design with UML* (2nd Edition ed., p. 46). UK: McGraw Hill Education.

Rainy. (2007). *Rainlendar - A customizable desktop calendar*. Retrieved October 25, 2007, from Rainlendar: http://www.rainlendar.net/cms/index.php

Reingold, E. M., & Dershowitz, N. (2001). *Calendrical Calculations: The Millennium Edition.* Cambridge: Press Syndicate of The University of Cambridge.

Ruston, J. (2004, December 10). *TiddlyWiki*. Retrieved October 25, 2007, from TiddlyWiki: http://www.tiddlywiki.com/

Santino, J. F. (1994). Calendar. *The New Book of Knowledge (Vol. 3, pp. 11-17)* . United States of America: Grolier Incorporated.

Soares, P. (2006, January 4). *PlasticCalendarPluginDoc*. Retrieved October 25, 2007, from TiddlyWiki 2.2.6 Addons: http://www.math.ist.utl.pt/~psoares/addons.html#PlasticCalendarPluginDoc

University of Chicago. (1993). Calendar. *Compton's Encyclopedia (Vol. 4, pp. 28-30)* . Chicago, United States of America: Compton's Learning Company.

Whitehead, J. (2007, July 22). *Welcome to WebDav Resources*. Retrieved October 26, 2007, from WebDAV Resources: http://www.webdav.org/

XMLHttpRequest. (2007, October 31). Retrieved November 1, 2007, from Wikipedia: http://en.wikipedia.org/wiki/XMLHttpRequest

Code References

Cross browser javascript calendar . (n.d.). Retrieved March 12, 2007, from dtmlgoodies.com: http://www.dhtmlgoodies.com/scripts/js_calendar/js_calendar.html

Hunlock, P. (2006, December 02). *Javascript Drag and Drop*. Retrieved March 20, 2007, from The Javascript Reference Series: http://www.hunlock.com/blogs/Javascript_Drag_and_Drop

Jenci, K. (2005, October 05). *JavaScript validate 'as you type'*. Retrieved September 05, 2007, from mredkj.com: http://www.mredkj.com/tutorials/validate2.js

Larson, J. (2005, May 4). *Clipboard Copy*. Retrieved September 15, 2007, from Jeffothy's Keyings: http://www.jeffothy.com/weblog/clipboard-copy/

Oster, J. (2007, August 26). *Javascript Toolkit*. Retrieved September 20, 2007, from Wingo Bay: http://www.wingo.com/jt_/index.html

Reingold, E. M., & Dershowitz, N. (2001). *Calendrical Calculations: The Millennium Edition.* Cambridge: Press Syndicate of The University of Cambridge.

Appendix A

Current Name	Historical Name	Meaning
Monday	Monandag	Day of the Moon
Tuesday	Tiesdag	Tiu, God of War
Wednesday	Wodnesdag	Woden, Ruler of Gods
Thursday	Thorsdag	Thor, Controller of Thunderbolt
Friday	Frigadag	Frigga. Wife of Woden
Saturday	Saeternesdag	Saturn, God of Agriculture
Sunday	Sunnandag	Day of the Sun

Table A1: Origins of English Week Day Names

Current Name	Historical Name	Origin
January	Januarius	Janus, God of Beginnings & Doorways
February	Februarius	Februa, Feast of Purification
March	Martius	God Mars
April	Aprilis	Goddess Venus (Aphrodite)
May	Maius	Goddess Maia
June	Junius	God Juno
July	Julius	Julius Caesar
August	Augustus	Julius Augustus
September	Septem	Seven
October	Octo	Eight
November	Nove	Nine
December	Decem	Ten

Table A2: Origins of English Month Names