

# The Web Pilot project

Fall 2004

by

Hugo Moen ( [hugo@itu.dk](mailto:hugo@itu.dk))  
Tor-Jarle Sagen ([tor@itu.dk](mailto:tor@itu.dk))  
Lars-Martin Tollefsen ([tollar@itu.dk](mailto:tollar@itu.dk))

Project supervisor:  
Rasmus Lerchedahl Petersen

URL for Web Pilot and WAP:  
<http://www.itteam.no:8080/webpilot>  
<http://www.itteam.no:8080/webpilot/wap>



The **IT** University  
of Copenhagen

MSc SWU- Fall 2004

Title: The Web Pilot  
4 week project, 7 ECTS

Pages in report :32  
Pages in appendix:95

## Foreword

This report has been written in December 2004 for a four-week project at the IT University of Copenhagen under the supervision of Rasmus Lerchedahl Petersen. The project group consisted of Hugo Moen, Tor Jarle Sagen and Lars-Martin Tollefsen. During the project, we developed a web based event calendar called the Web Pilot, which can be used for a small research department of a small company. The Web Pilot let each employee log inn and access the calendar through a browser. The user can then view the calendar, create new appointments involving other people and groups. People who have been invited to a meeting are then able to respond to an invitation.

This rapport is the result of this work, where we have documented the implementation of the Web Pilot. We have also looked into concurrency, security, and support for different client types such as WAP. Web pilot has been tested extensively with black box method, and no errors has been found at this stage.

The goal with this project was to get a better insight in web development using MySQL and PHP, and look at problems that occur in a multi-user three-tier architecture.

Accessing web pilot can be done using the information found on the front page of this report. We have created an admin user with full privileges to the calendar. Sensor should login as admin and create a new user for himself in the admin view, so he can receive e-mail updates for changing password and cancelling events.

IT university of Copenhagen 14 December 2004

---

Tor Jarle Sagen

---

Lars-Martin Tollefsen

---

Hugo Moen



# Contents

Foreword.....	2
Contents .....	3
Problem description .....	4
Analysis and design .....	5
Main Use Case.....	5
Implementation.....	6
Login .....	6
Administration.....	7
User profile.....	7
Event .....	9
Calendar .....	10
Help guide .....	11
Layout.....	12
Database design .....	13
Normalization .....	14
First normal form .....	14
Second normal form.....	14
Third normal form.....	15
Entities.....	15
Support for different client types.....	18
Internet browsers.....	18
Mobile phone with WAP support.....	18
Design/flowchart.....	19
Screen Captures.....	20
Further improvements .....	20
Concurrency control.....	21
Concurrency in Web Pilot .....	21
Transactions.....	21
Read/Write locks .....	22
Deadlock.....	22
Security.....	23
Password .....	23
Session.....	23
Testing .....	24
Black box test .....	24
Firewall problem.....	26
Technical description of the program .....	27
Session .....	27
Time & Date .....	27
Reminders.....	28
Uploading images.....	28
System environment.....	30
Conclusion .....	31
Reference .....	32
URL reference .....	32
Figure reference.....	32
Supplemental literature.....	32
Appendix.....	33



## Problem description

The aim of the project is to develop a web based calendar for a small research department or a small company. Each employee should be able to access the calendar through a browser of her choice. Having logged on, the user should be able to view her calendar, or create new appointments, possibly involving other people. However, it should not be possible to invite a person to a meeting if the new meeting collides with another appointment for that person. People who have this way been invited to a meeting should be able to respond to the invitation.

The work on the project should result in an implementation of the calendar and a report documenting the work. The report should describe concurrency problems that could occur in the calendar, and how these problems are taken care of. Also how a calendar can be shown in different clients like WAP. Testing of the product is an important aspect of the report.

A detailed project description can be found at the project home page. (URL 1)

## Analysis and design

In this part of the report we describe the analysis and design of the web pilot. Where we first show the main use case and then the implementation.

### Main Use Case

In the use case model below we show the relationship among the user and the functionality within the system.

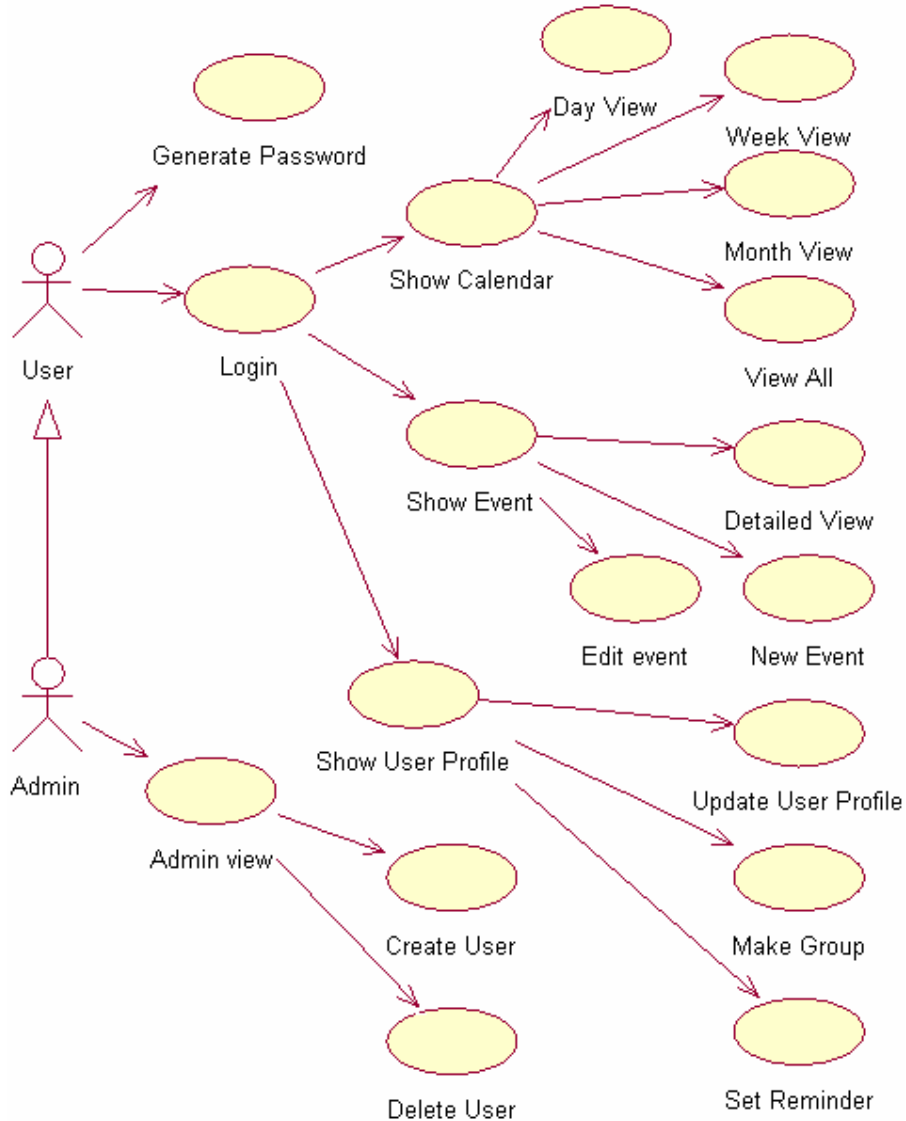


Figure 1 Web Pilot Use Case

## Implementation

In this section we describe how we have implemented the use cases in our solution.

### Login

To prevent the system from being accessed by unauthorized users, a login function is included. At the login the user is asked to state the username (e-mail) and a password. A user is unique identified by an e-mail address. The typed inn password is encrypted and compared to the password in the database. If the password is accepted the user is logged on to the web pilot. A set of session variables for the user is then registered in the login.php file. The login.php file is included in each file, so that session is included in the whole system. This helps the system to hold track of the user, and see if the user are logged in or not. All the profile information for the user is to be found in the session, this makes the system to have fewer queries to the database.



Log in	
Username(e-mail):	tollan@netcom.no
Password:	*****
<a href="#">Forgot password?</a>	<input type="button" value="Log in"/>

Figure 2 Login screen

If the user has forgotten the password, a new one can be generated and sent by mail to the user. This is of course only possible if the users e-mail is registered in the database.

## Administration

Web pilot users can either be a user or an admin. An administrator will have an extra link under his/hers profile called "Manage users". This view will list all the current registered users of the web pilot. An administrator will be able to add, edit or delete users.

[Web Pilot event calendar](#)

Users 13 December 2004, 11:12 am

<ul style="list-style-type: none"><li>- Calendar</li><li>- Your profile</li><li>- Your groups</li><li>- Admin users</li><li>- log out</li></ul>	first name	last name	email	user	Create user	
	Adrichem, Yvette		amorakus@hotmail.com	Change	user	delete
	Brønstad, Per Lasse		timon70@excite.com	Change	user	delete
	Haga, Marius		marius@itteam.no	Change	admin	delete
	Karlsen, Rune		aeldarin@hotmail.com	Change	user	delete
	Kvam, Annette		annette@itteam.no	Change	user	delete
	Moen, Hugo Michal		hugo@itteam.no	Change	admin	delete
	Moen, Ivar		ivar@husservice.no	Change	user	delete
	Petersen, Rasmus Lerchedahl		rusmus@itu.dk	Change	user	delete
	Sagen, Tor Jarle		tor@itu.dk	Change	admin	delete
	Tangen, Ove		ovet@yahoo.com	Change	user	delete
	Tollefsen, Lars-Martin					

Figure 3 Admin view

To add a user an admin must supply a name, an e-mail address and a status (user/admin) to the new user. The user is then added to the database and an e-mail is sent to the new user with username (e-mail), password and a URL to the service.

An administrator can also change the users e-mail address and status, or delete a user from the system. When a e-mail is changed an e-mail will be sent to that user.

## User profile


A registered member of the web pilot can view change hers/his user profile. A new user will only have a name, an e-mail address and a password in the profile as default. When logged on a user can update/change the profile. In addition to basic information like name, address, phone number etc, a user has the possibility to change the password. If the user changes her password, the new password will be sent to her e-mail. Three features of the user profile that needs to be mentioned are the setting of reminders, uploading of pictures and making user groups.



**User profile**

- Calendar
- Your profile
- Your groups
- Admin users
- Log out

First name:	<input type="text" value="Lars-Martin"/>
Last name:	<input type="text" value="Tollefsen"/>
Street	<input type="text" value="Sofievej 13"/>
Postal code	<input type="text" value="2900"/>
Postal town	<input type="text" value="Hellerup"/>
Homepage	<input type="text" value="www.tollefsen.tk"/>
Email	<input type="text" value="tollar@netoom.no"/>
Change of e-mail sets new user login.	
Office number	<input type="text" value="1"/>
Phone number	<input type="text"/>
Mobil number	<input type="text" value="+47 922 40351"/>
Privileges:	( admin )
Change password:	<input type="text"/> min. 6 characters
Repeat password:	<input type="text"/> min. 6 characters
<input type="button" value="Update"/>	



Upload Image

Set your reminder ☺

No reminder ▼

**Figure 4 User profile**

The remainders option gives the user a possibility to be reminded a given numbers of minutes before an event is to take place. As default no reminder is set for the user. The system checks once every minute to see if the user has set the remainders option. With the remainders option set the user will get a warning informing on start time and title of the upcoming event. The warning will be visible on the top of the page, regardless of which page within the system the user is watching. From the chosen minutes before an event start until reaching start time of the event, it will be visible on the top of the screen.

Web Pilot event calendar

**EVENT REMINDER: Frokost is starting at 12:00** [View event details](#)

[Day](#) | [Week](#) | [Month](#) | [All events](#)

[New event](#) | [Log out](#)

**Figure 5 Reminder alarm**

The feature “Upload image” will let the user add a personal picture to the user profile. The user browses his image directory and selects an image of the type .jpg, .bmp or .gif. It will then be uploaded and presented in the user profile.



The function “Your groups” lets users form their own groups of web pilot members. This function is added so that users can invite groups to events, instead of having to invite each person. A group is created by giving it a name, then users is added to the group. Groups can be edited or deleted.

**Web Pilot event calendar**

**Figure 6 Group screen**

**Event**

A user can add a new event, edit an event or delete an event from the database. A new event is added to the database through a form. All values in the form menus are built using PHP. The form is reloaded when needed so only valid values can be selected. E.g. if a user select 2005 as start year, and February as month, only the dates from 1 to 28 is listed in the date menu. If the form is reloaded due to the above mentioned reason, or because the user has left out a mandatory field, the form “remembers” the entered values.

**Web Pilot event calendar**

**Figure 7 Submit event screen**

When a user has submitted a new event with valid values in all fields, there is a check for overlapping events involving the creator of the event. If there is overlapping events, the user will be informed. The user has now the choice between



skipping adding the event, or forcing the event to be added despite of the overlapping event(s).

The “force event” option is not a part of the requirements for the web pilot. We have added this option because we feel that it should be the user, not the system that is controlling the decisions. Let’s say we have a situation where a user has an event running over several days. He or she should still be able to book another event within the time limit of this event.

Another possibility is to make events “public”, meaning that all users will be able to view the event in their calendar view. The event that is public has a “(public)” tag behind the title name, showing that it’s public. Events are by default private.

If the event only involves the creator, the process of adding the event is finished. If it involves other participants, it is now time to invite them to the event. Participants are added to the event one at a time. There is a check if participants have overlapping events before they are invited to the event. A user can not be invited to an event if there is overlapping events. If a group is invited to the event and some group members are busy, only the available ones are invited to the event.

If an event gets deleted by the creator of the event, an e-mail will be sent to all participants informing that the event is cancelled.

## Calendar

The idea on how to make a calendar was found in a tutorial (URL 2), that we used as a starting point for the calendar.

Calendar is the view where the user can get an overview of the events. The three main views the user can choose between is day, week and month.

### Web Pilot event calendar

Day | Week | Month | All events New event | Log out

Dec 2004 Monday, December 13, 2004 ◀ Prev day - Next day ▶

M	T	W	T	F	S	S
29	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

Today is:  
Mon, Dec 13, 2004

Welcome Lars-Martin  
- Edit user profile  
- You have 1 open requests

Mon 14:00 - Mon 14:59	<b>Project meeting</b> Read through report and look for errors and things to add.. <a href="#">more</a>
Mon 18:00 - Mon 19:59	<b>Christmas dinner (public)</b> Dinner with the members of the Web Pilot Group. <a href="#">more</a>

© Web Pilot Team 2004

Figure 8 Calendar day screen

The day view is the default view which is presented for the user when logging on. In the day view all events for the current day is listed with start time, end time, title and



the first part of the description. Clicking the title will lead to a page showing all the details about the event. There is next and previous links allowing the user to navigate among the days.

The week view does exactly what the name indicates, it shows all days in the week from Monday to Sunday. The functionality is almost identical to the day view.

The month view gives the user an overview of all events in the chosen month. Due to limited space only the project title is shown in this view. The today date is shown in a lighter colour than the rest of the days. In addition to prev and next links, we also have included a menu allowing the user to quickly jump to a month.

There are also views for “open requests” and “all events”. The open request view lists the events the user has been invited to, but not replied to yet. In this view the user has the possibility to confirm or decline events. Confirming an event will make it show up in the user’s calendar. The view “All events” list all events the user has confirmed, and all public events. We thought it would be handy for the user to view all upcoming events without having to navigate among days, weeks or months.

## **Help guide**

We have implemented a few help guides as a question mark that can be clicked, and then shows a popup window with more information about the topic next to the question mark. Currently there are help guides on the public choice in new events, set reminder under the user profile and force event choice if you get an overlapping event when adding a new event. The help guide uses JavaScript to popup the help window, and will not work if JavaScript has been disabled in the internet browser.

## Layout

### Web Pilot event calendar

The screenshot displays a web interface for an event calendar. At the top is a **HEADER** bar with navigation links: Day | Week | Month | All events, New event | Log out. Below the header is a navigation bar for **Monday, December 13, 2004**, with links for Prev day and Next day. On the left is a **MINI CALENDAR** for Dec 2004, showing the current date (13) and a 'Today is: Mon, Dec 13, 2004' message. Below this is a user welcome message: 'Welcome Lars-Martin' with links to 'Edit user profile' and 'You have 1 open requests'. The main area is the **EVENT VIEW**, which lists two events for the day: 'Project meeting' (14:00-14:59) and 'Christmas dinner (public)' (18:00-19:59). The footer contains the copyright notice '© Web Pilot Team 2004' and the **FOOTER** label.

Figure 9 Layout of day view

Our view is divided into header, footer, calendar view and event view. This is our main view, where the user sees the day's event. The footer and header have links to the other calendar views, and link for making new event.

We use the Arial font on all our pages, because it's a clear font that most web browsers support. Through the web pilot we have used the same colour design as above, that we have consistently used through the application. For important messages we used a red coloured font that shows good.

## Database design

images
img_imgtype
img_imgdesc
img_imgfile
img_imgsize
img_lastmod
img_imgbin

groups
group_name
owner

events
description
title
start_time
end_time
location
creator_event
accept_request
public

web_user
user_password
first_name
last_name
street
postal_code
postal_town
email
office_number
phone_number
mobile_number
homepage
reminder
privelige_type

Figure 10 First table selection

After an examination of the requirements we come up with these four tables. But before we can use the database these tables need to be organized. So we need to normalize the database to reduce problems with consistency by reducing redundancy. We can see a lot of redundancy problems in the above tables. The events table has entities like creator\_event and accept\_request that will give a lot of redundancy for the other entities in the table, because many users can have, or be part of same events. The same goes for groups table, where many users can have a group, or be part of groups.

The normalization of the database is a multistep process, and as a minimum we want to normalize to the third normal form. Below we show the steps taken to get there.

## Normalization

As we see in the EER diagram figure below we have normalized our data model to the third form. We did the normalization forms step by step through the forms until the third normal form.

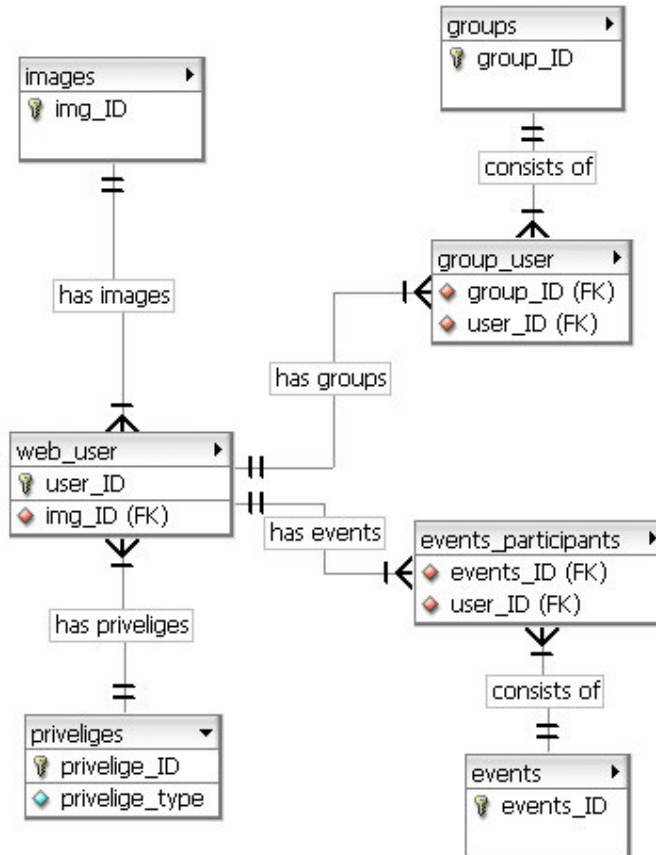


Figure 11 Database in third normal form

This is the EER diagram for our system, showing the relationships and PK and FK in the entities.

### First normal form

This form means that we should eliminate duplicate columns from the same table. That means that we should create separate tables for each group of related data and identify each row with a unique column (the primary key).

### Second normal form

In this form we want to remove subsets of data that apply to multiple rows of a table and place them in separate rows. That means that we should create relationships between these new tables and their predecessors through the use of foreign keys. Here we have tried to reduce the amount of redundant data in a table by extracting it, and placing it in new tables and creating relationships between those tables. As shown in the EER diagram we can see that we have just done that with the `web_user`



and groups, where we have extracted the data out in the group\_user table. The same procedure we have done with events\_participants.

### Third normal form

This form requires that we have met the requirements of 1NF and 2NF, where we remove columns that are not fully dependent upon the primary key.

We can see that we could have normalized more in the web\_user table, placing postal\_code and postal\_town out in their own table. But we mean it is not necessary for this application, because it will just create minor redundancy.

### Entities

Here all entities with their attributes are listed for the web pilot MySQL database.

events							
ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
events_ID	INTEGER(10)	PK	NN	UNSIGNED		Events is, auto incremented id number	AI
description	TEXT					Event description	
title	VARCHAR(45)					Event title	
start_time	BIGINT(20)					Event start time	
end_time	BIGINT(20)					Event end time	
location	VARCHAR(45)					Event location	
public	INTEGER(1)		NN	UNSIGNED		1 is public, and 0 is not public	
IndexName	IndexType	Columns					
PRIMARY	PRIMARY	events_ID					

events_participants							
ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
events_ID	INTEGER(10)		NN	UNSIGNED		Foreign Key relation from events	
user_ID	INTEGER(10)		NN	UNSIGNED		Foreign Key relation from web_user	
creator_event	TINYINT(1)					1 = User created the event, 0 = User did not create event	
accept_request	TINYINT(1)					1 = Confirmed request, 0 = Open request, -1 = Declined request	

group_user							
ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
group_ID	INTEGER(10)		NN	UNSIGNED		Foreign Key relation from groups table	
user_ID	INTEGER(10)		NN	UNSIGNED		Foreign key relation from web_user table	
owner	TINYINT(1)			UNSIGNED		1 = owner of group, 0 = Not owner of group	



<b>groups</b>							
ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
group_ID	INTEGER(10)	PK	NN	UNSIGNED		Primary Key of group ID.	AI
group_name	VARCHAR(45)					Name of the group	
IndexName	IndexType			Columns			
PRIMARY	PRIMARY			group_ID			

<b>images</b>							
Image base for user pictures							
ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
img_ID	INTEGER(10)	PK	NN	UNSIGNED		Primary key for image	AI
img_imgtype	VARCHAR(100)					Support gif, jpg and bmp pictures	
img_imgdesc	INTEGER(10)					Static set to 1. We use no picture description	
img_imgfile	VARCHAR(255)					Filename of the picture	
img_imgsize	INTEGER(10)					Bytesize of the picture	
img_lastmod	VARCHAR(30)					Last modified picture date	
img_imgbin	LONGBLOB					Physical picture in binary form	
IndexName	IndexType			Columns			
PRIMARY	PRIMARY			img_ID			

<b>priveliges</b>							
ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
privelige_ID	INTEGER(10)	PK	NN	UNSIGNED		Primary key for privelige	AI
privelige_type	VARCHAR(20)					Type of privelige. (user, admin)	
IndexName	IndexType			Columns			
PRIMARY	PRIMARY			privelige_ID			



<b>web_user</b>							
<b>ColumnName</b>	<b>DataType</b>	<b>PrimaryKey</b>	<b>NotNull</b>	<b>Flags</b>	<b>Default Value</b>	<b>Comment</b>	<b>AutoInc</b>
user_ID	INTEGER(10)	PK	NN	UNSIGNED		Auto Incremented user id.	AI
img_ID	INTEGER(10)		NN	UNSIGNED		Foreign Key relation from image table	
privelege_ID	INTEGER(10)		NN	UNSIGNED		Foreign Key relation from priveleges table	
user_password	VARCHAR(45)		NN			Password field, it's encrypted with MD5 encryption	
first_name	VARCHAR(45)					First name	
last_name	VARCHAR(45)					Last name	
street	VARCHAR(45)					Street name	
postal_code	VARCHAR(20)					Postal code	
postal_town	VARCHAR(45)					Postal town	
email	VARCHAR(45)		NN			Email is required, it's used as unique identification of the user.	
office_number	VARCHAR(20)					Office room number	
phone_number	VARCHAR(20)					Phone number	
mobile_number	VARCHAR(20)					Mobile number	
homepage	VARCHAR(100)					Homepage, as URL link	
reminder	INTEGER(10)			UNSIGNED		Set a reminder in minutes, for alarm to go off	
<b>IndexName</b>		<b>IndexType</b>		<b>Columns</b>			
PRIMARY		PRIMARY		user_ID			



## Support for different client types

### Internet browsers

We have used 3 different browsers to test web pilot in. Internet Explorer, Opera and Mozilla Firefox. Web pilot works in all internet browsers, but there are minor differences of how the pages are presented.

### Mobile phone with WAP support

Today most people have new mobile phones that supports the WAP(Wireless Application Protocol) protocol, and GPRS(General Packet Radio Services). This makes the use of WAP more actual today than a few years ago. This is because you can be online all the time, with a much higher speed than just a few years ago. Therefore it should be possible for the user to get today's events on the phone, when she is on the run.

### Installing WAP support on a server

Before we could start developing on a WAP solution for our Web Pilot, we had to set up the server so it could understand the data types required for WAP. The datatype **"text/vnd.wap.wml wml"** must be added to the web server configuration files. This will force the server to send out the mime-type "text/vnd.wap.wml" with every .wml page in the directory. The correct way to configure the web server is to add the following 5 mime-types to the server configuration file.

Content	MIME type	Extension
WML source	text/vnd.wap.wml	Wml
Compiled WML	Application/vnd.wap.wmlc	Wmlc
WMLScript source	text/vnd.wap.wmlscript	Wmls
Compiled WMLScript	Application/vnd.wap.wmlscriptc	Wmlsc
Wireless bitmap	image/vnd.wap.wbmp	wbmp

**Table 1** WAP datatypes

The WAP device' communication with the web server is in principle much the same as that between a web server and a traditional pc web browser. It just requires an extra step that involves the transfer of information by a WAP gateway. WAP gateways serve as intermediaries between the client-based wireless browser and an information server.

This process is illustrated below:

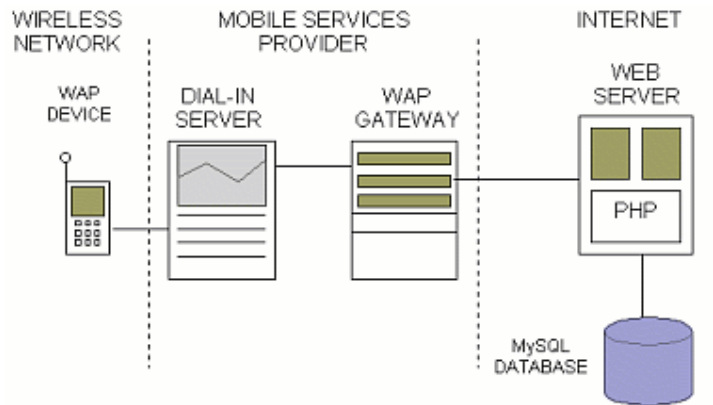


Figure 12 WAP communication (URL 3)

### WAP and WML a short description

The WAP standard is based on Internet standards (HTML, XML and TCP/IP). The purpose of this standard is to show internet contents on wireless clients like mobile phones. It consists of a WML (Wireless Markup Language) language specification, which is defined as an XML 1.0 application.

WML stands for Wireless Markup Language, this markup language are inherited from HTML, but WML is based on XML, but much stricter in use than HTML. WML is used to create pages that can be displayed in a WAP browser. Pages in WML are called Decks and are constructed as a set of Cards.

### Design/flowchart

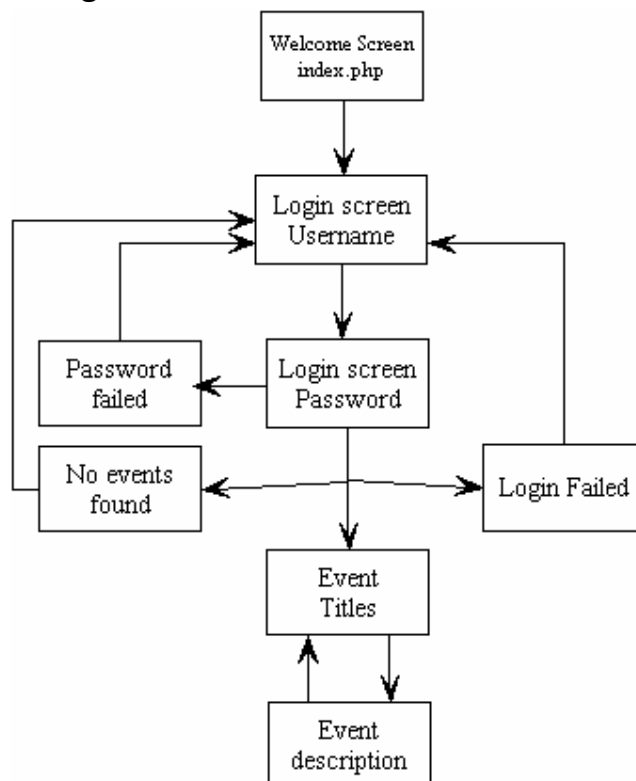


Figure 13 WAP flowchart

The WAP implementation for showing today's plan is quite simple. The user login with her e-mail address and password. If there are any events, the titles will be shown and can be selected to get a full description with start and end time. It's also possible to check for events by selecting the day ahead or day before the current day, by selecting the day links above the listed events. In case of failures like entering wrong password or connection problems to the database, a message will appear and then send the user back to the login screen..

## Screen Captures

Here we see the different stages when using the WAP phone emulator.



Figure 14 Login screen



Figure 15 Today's Event



Figure 16 Description of event

In Figure 14 we show the login screen where the user enters his e-mail and password. Next is the listing of today's event with links to the previous and next day, so the user can browse for events. Last screen shows a detailed view of the selected event, showing start and end time of the event.

## Further improvements

The user might also want to make a new event from the mobile, by writing in new event and times.

## Concurrency control

The Web Pilot is a multi user concurrent system where several users or processes at the same time may try to use the same resources. The issue of concurrency control arises when you want to perform a sequence of actions isolated from other actions in a concurrent system, which is without interference.

### *Concurrency in Web Pilot*

In web pilot there is one place that has some critical areas, which is under the event creation, and want to add participants to the event.

One of the problems with creating events is adding users to your event. A person you select from the pull down menu to add to your event, could have been selected by another person just the moment before. The other problem is if you try to create an event yourself, and get invited to an event from another user simultaneously. These two problems show the concurrency problems the user faces. We have dealt with them by using table locking in our sql code. We have to lock all tables in use, because they are dependent of each other in the query.

Below we show how PHP/MySQL achieves this.

```
LOCK TABLES events WRITE, events_participants WRITE, web_user
WRITE group_user WRITE;
...
sql code here
...
UNLOCK TABLES;
```

We tested locking of tables by using a sleep function in the PHP script so the lock could stay locked on a table longer, and then tried to read from the locked table from another client. It was not possible to get result from the locked table until the sleep function was finished. We had to put a sleep function in the PHP script because when the PHP scrip terminated, it releases locked tables automatically.

When we use locking on tables, we block connection for another connection to the sql server that requires a conflicting lock type on the records locked by the first connection. This causes the second connection to wait until the first connection releases its locks. It's important that this waiting is as short as possible, or locking might be a bottleneck to our system. We have made the use of our locks to be as short as possible to reduce blocking, for best performance.

To achieve concurrency control there are different approaches we could use, that I mention briefly below.

### *Transactions*

To enforce concurrency control we got to treat SQL-queries automatically, as a single unit of work. Either all queries in a group are applied to the database, or none of them are. This is part of the ACID properties that must be satisfied for transactions. Their meaning is:



Atomicity	All or nothing
Consistency	preserves constraints
Isolation	serial equivalence
Durability	changes are persistent

In my MySQL transactions are wrapped with BEGIN and COMMIT statements. The transaction manager in MySQL, InnoDB handles most of the ACID properties. Atomicity, Isolation and Durability is guaranteed by InnoDB, while the programmer has to handle Consistency.

### ***Read/Write locks***

MySQL has to prevent one client from reading a piece of data while another is changing it. MySQL performs this lock management internally in a way that is transparent much of the time for the user. What happens is that the system deals with concurrent read/write access with a system that consists of two lock types, read locks and write locks. The concept is that read locks on a resource are shared. Many clients may read from the database at the same time and does not interfere with each other. Write locks are exclusive, because it's safest to have only one client writing to the database at a given time to prevent all reads when the client is writing. This is because a single writer could make any changes to the database, even deleting it.

### ***Deadlock***

When multiple transactions obtain locks, there is danger of getting into a deadlock condition. Deadlock happens when two transactions attempt to obtain conflicting locks in a different order. To prevent this MySQL uses InnoDB which is a storage engine designed for transaction processing. It has a deadlock detection system that is default set to 50 seconds. But systems that use transactions should handle deadlock itself, and possibly retry to make the transactions.

## **Security**

The security of the system is maintained through the use of passwords and sessions.

## **Password**

When a user is added to the system, a password is generated with a call to the randPass function.

```
$kode = randPass();
```

The randPass function returns a random generated password between 6 and 9 characters long.

```
function randPass()
{
    $start = rand(0,6);
    return substr( md5(rand(1111,99999)), $start , rand($start+6,$start+9));
}
```

The password is then encrypted with the md5 algorithm, as shown below, before it is stored in the database. The only decrypted version of the password that exists is the version that is sent to the user in an e-mail.

```
$user_pw = md5($kode);
```

## **Session**

The log in file with the session variables is included in every file in the system. This means that it is not possible, without being logged in, to enter a page in the site structure by typing the URL in the browser. The log in page will pop up, demanding a valid e-mail and password for the user. There is also added a feature that prevents ordinary users from entering the admin page where users is added, edited or deleted.

## Testing

We have done a thoroughly testing of web pilot, and spend a great amount of time to quality check our code. Below we have used a black box approach and checked important functionality in the system.

### **Black box test**

The administrator user is added in the database from before. We are testing on creating and administrating users with e-mail and password issues. Administrator has user id 1 and users with id 2 and 3.

Nr.	Name	Problem	Expected	Failed/ok
1	Add user 2	Add 1 user and generate password and sending mail.	should get e-mail with login information	OK
2	Add user 3	Add another user and generate another password and sending mail.	should get e-mail with login information and different password	OK
3	Remove user 3	Remove user 3	The user should disappear from the admin user list.	OK
4	Add user 3	Re add user 3	should get e-mail with login information and new password	OK
5	Password	To get new password for user 2 form login	User 2 get e-mail with new password	OK
6	e-mail	Give new e-mail to user 3 from admin.	User get e-mail at the new address with new password	OK
7	Profile	User 2: Update profile, logout and login again. Check profile.	The new info should be there.	OK
8	Profile	User 3: Make new password, logout and login again with new password.	Should be able to login again, and get e-mail with new password.	OK

Testing user 3 with creating groups.

Nr.	Name	Problem	Expected	Failed/ok
9	Add group	User 3 make a new group called "test group1"	"test group 1" should show up	OK
10	Add users	Add user 1 and user 2 to group "test group1"	Users name should show up under the group "test group 1"	OK
11	Delete user	Delete user 2 from "test group 1"	User 2 should disappear form the group.	OK



12	Delete group	Delete whole group "test group 1"	The group should disappear from the site.	OK
13	Add group	Add group "test group 2", and add user 1 and 2 to the group.	The site should show with the new group and their new users.	OK

User 2 is making events and test for the rolls are working.

Nr.	Name	Problem	Expected	Failed/ok
14	Add event 1	Missing argument: title	Title missing	OK
15	Add event 2	Missing argument: description	Description missing	OK
16	Add event 3	Missing argument: location	Location missing	OK
17	Add event 4	Add event from 4.jan.16.00 to 4.jan 12.00	End time is earlier then start time.	OK
18	Add event 5	Add event from 4.jan.12.00 to 4.jan 16.00	Event inserted in the database - OK	OK
19	Finish event 5	Press the finish button.	The event will show up in full size.	OK
20	Delete event 5	Press delete button on the event detail side.	Return to day view and the event is no longer there	OK
21	Add event 6	Add event from 4.jan.12.00 to 4.jan 16.00	Events shall be added	OK
22	Add event 7	Add event from 4.jan.16.00 to 4.jan 20.00	Events shall be added	OK
23	Add event 8	Add event from 4.jan.10.00 to 4.jan 13.00	Overlapping events: events 6	OK
24	Add events 9	Add event from 3.jan.10.00 to 5.jan 13.00	Overlapping events: events 6 and event 7	OK
25	Add events 10	Add event from 4.jan.19.00 to 5.jan 13.00	Overlapping events: event 7	OK



### Testing multiple users on events.

Nr.	Name	Problem	Expected	Failed/ ok
26	Add events user 1	User 1: add event from 4.feb.10.00 to 4.feb 15.00	Events shall be added	OK
27	Add events user 2	User 2: add event from 6.feb.10.00 to 6. feb15.00	Events shall be added	OK
28	Add events user 3	User 3: add event from 6.feb.10.00 to 6. feb12.00	Events shall be added	OK
29	Add user	User 3 shall add user 1 to this event.	User shall be added	OK
30	Add user	User 3 shall add user 2 to this event.	User is busy	OK
31	Remove user	User 3: Remove user 1 from list	User shall disappear from list.	OK
32	Add group	User 3: Add group "test group 2" to list.	User 2 is busy and user 1 shall be added to list.	OK

We have gone through these black box tests several times during the development to make sure that changes in code has not inflicted on the expected test results. A great time has been spent on walkthrough of code to remove syntax and logical errors.

## Firewall problem

When deploying our web pilot onto the server, one of us got some strange problems using web pilot. When a link or button got pressed, he had to login again. This is typically a cookie problem. We found out that it was the Kerio Personal Firewall that had blocked the use of cookies. The "Filter foreign cookies" was then unchecked, and web pilot worked normally again. We have tested this on other firewalls like the windows build in firewall, and "tiny firewall" with no problems. This shows that some firewalls might be too strictly setup, to block cookies. This will cause the session to be broken. A solution to this would be to change the php settings to send session ID in the URL, but this is not very safe. The session string could be copied or bookmarked, so unwanted person could access the site. That's why we choose to go with the default setting, and store the session in a cookie.

## Technical description of the program

This section contains an explanation to details in the source code. We emphasize on explaining "clever constructions" and leaving out trivial code. A complete listing of the code, with comments, can be found in the appendix.

### **Session**

The user session variables is registered when the user log on, and stay with the user until logging out or closing the browser. Sessions in PHP are driven by a unique session ID, a cryptographically random number. The session ID is generated by PHP and stored on the client side for the lifetime of a session. It can be either stored on a computer in a cookie, or passed along through URLs.

This allows you to register so-called session variables. The content of this variables are stored at the server.

There are 4 step of using session.

- Starting the session – a user enters the log in page.
- Registering the session variables – the user has supplied a valid log in
- Using session variables – throughout the lifetime of the session
- De registering variables and destroying the session – logging out or closing the browser.

### **Time & Date**

To handle the time and date issues in the program, we do extensively use of Unix timestamps. A Unix timestamp is a 32-bit integer containing the number of seconds since midnight, January 1, 1970. To handle the timestamps we use the two PHP functions mktime() and date(). The mktime function is used to create the timestamps. The mktime function returns a timestamp, dependent on the parameters we supply. Calling the mktime function without parameters will return current values. The date function is used to convert the timestamps to "normal" time/date before presenting it for the user.

Example from day.php:

Here we are combining the mktime and date function to give us the current day, month and year, not caring about the time.

```
$date = mktime(0,0,0,date('m'), date('d'), date('Y'));
```

Now that we have a Unix timestamp we can easily get the information we want using the date function.

```
$day = date('d', $date); //day as a 2-digit number  
$month = date('m', $date); //month as a 2-digit number  
$year = date('Y', $date); //year as a 4-digit format
```



Having the current values, we can get the next day by just adding 1 to the \$day variable, when making a new timestamp.

```
$this_next_link .= mktime(0,0,0,$month,($day + 1),$year);
```

We must of course take into consideration issues like if it is the 1.January or 31.December in which we must change the \$year variable.

## ***Reminders***

The reminders option uses a page that updates itself once every minute. A query is then made to the database to check to see if the reminders option is set. The following will happen if this option is set through the user profile.

First we create a Unix timestamp for the current time.

```
$current_time = mktime();
```

Then we make variables for hour, minute, second, month, day and year.

```
$hour = date('H', $current_time);
```

```
$minute = date('i', $current_time);
```

We then add the user set reminder option to the minute variable.

```
$minute = $minute + $reminder;
```

Then make a new timestamp for the reminder limit.

```
$reminder_limit = mktime($hour,$minute,$second,$month,$day,$year);
```

We now have enough information to perform a query to the database, to see if the user has any events that needs reminding. The query returns the events that are within the user's reminder option. The output of the reminder will be visible for the user through an iframe (an inline frame that contains another document).

## ***Uploading images***

To get an image from a computer to the web pilot, there is a special kind of form to be used. The form has to be set as **ENCTYPE="multipart/form-data"** and there has to be used an input type called "file". This form set makes the user able to browse through the computer after a file to be uploaded. The file will be temporary saved in a folder under PHP, on the server. Here we check what kind of file type it is. The system is made for only accept .jpg, .gif and .png files. Since we only wanted small images, the size of the picture will be reduced before it gets saved to the database. Since we are saving images in the database, there will be needed some techniques to show them on a web page. To show an image that is a result from a query, we have to set the header information to the image content type. The content type is then one of the supported picture types. This must be done first in the file. Therefore we are using a separate file (getimg.php), that contains the raw data of the image that is shown on the screen.



```

// printout of the result for the function imgLink
echo imgLink($session_img_ID);

// The function return an img tag with source to the file getimg.php
function imgLink($imgID) {
    // Writes the HTML-trag that loads the picture
    if ($imgID != 0){
        return "<img src=\"../inc/getimg.php?img_ID=$imgID\" border=0>";
    }else return "&nbsp;";
}

//Dynamic file that gets header information after what content image type, and the
//raw image data from the database.
<?
if (!$img_ID) $img_ID = "1";
include ("function.php");
mydb_connect();
$img = mysql_fetch_row(mysql_query("SELECT img_imgtype, img_lastmod,
img_imgbin FROM images WHERE img_ID=$img_ID"));
header("Content-type: $img[0]");
header("Last-Modified: $img[1] GMT");
echo $img[2];
?>

```

## System environment

Web Pilot was programmed using PHP v.5.0.2 (URL 4). MySQL v.4.1.7 (URL 5) was used for the database. Web pilot is running on a Microsoft IIS 4.0 web server, on a Windows 2000 operating system.

We also installed extensions for PHP called GD2, to be able to use new features with images in php. In PHP v.5.0.2 there was no support for MySQL so we needed to install the .dll files for MySQL support in the PHP ext directory by adding their location to the php.ini file, in the windows system directory.

## Conclusion

This project has resulted in a fully functional event calendar, ready to be implemented in a small research department or a small company. We are quite satisfied with the result both on design and functionality. We have carried out an extensive black box test to ensure that the program works properly. But with this projects limited timescale we can't guarantee that the product is totally free from minor bugs.

Looking back now that the project is almost over, there is things that we would have done different starting over again. First of all we see the need for a longer period planning the project before we start on the actual coding of the system. The insufficient planning has lead to extra work, needing to alter the database and the scripts as new problems have arisen. We have also learned that it is no point in complicating things too much, but to rather code after the KISS (keep it simple stupid) principle.

There is always room for improvements and new functionality. If the product where presented for a large test group, we are sure that they would have given us valuable feedback to use to improve the product.

We feel that we have learned a lot from this project, and that our web programming skills has improved considerably. We have added several features to web pilot additional to the basic requirements, so with our new acquired skills we think we could have taken on more projects like this.

## Reference

### **URL reference**

- (URL 1) <http://www.itu.dk/people/mogel/WebPilot/>
- (URL 2) <http://www.phpfreaks.com/tutorials/83/0.php>
- (URL 3) <http://www.zend.com>
- (URL 4) <http://www.php.net>
- (URL 5) <http://dev.mysql.com/>

### **Figure reference**

Figure 1 Web Pilot Use Case .....	5
Figure 2 Login screen .....	6
Figure 3 Admin view .....	7
Figure 4 User profile.....	8
Figure 5 Reminder alarm .....	8
Figure 6 Group screen.....	9
Figure 7 Submit event screen .....	9
Figure 8 Calendar day screen .....	10
Figure 9 Layout of day view .....	12
Figure 10 First table selection .....	13
Figure 11 Database in third normal form.....	14
Figure 12 WAP communication (URL 3).....	19
Figure 13 WAP flowchart .....	19
Figure 14 Login screen .....	20
Figure 15 Todays Event .....	20
Figure 16 Description of event.....	20

### **Supplemental literature**

Balling, Derek J and Jeremy Zawodny(2004): High Performance MySQL. Publisher: O'Reilly.

The W2 homepage.

<http://www.itu.dk/people/jcg/WP/Conc/www/printer.php>

MySQL Manual | 1.5.5.3 Transactions and Atomic Operations

[http://dev.mysql.com/doc/mysql/en/ANSI\\_diff\\_Transactions.html](http://dev.mysql.com/doc/mysql/en/ANSI_diff_Transactions.html)

Yank, Kevin(2001):Build your own Database Driven Website Using PHP & MySQL  
Publisher: SitePoint Pty. Ltd.

Welling, Luke and Laura Thomson(2003):PHP and MySql Web Development.  
Publisher: Sams Publisher.



# Appendix

Appendix 1 Source code

