

# Exam, Web Publishing with Databases, Spring 2004

by Arne Glenstrup and Martin Elsman

---


- *All aids are permitted, except computers.*
- *The exam is a written exam, four hours, and is evaluated on a scale from 0 til 100 percent.*
- *The exam consists of five exercises, all of which should be solved.*
- *One word of advice: Read through the entire text before starting answering any of the exercises.*

## Introduction

In this exam you are asked to implement parts of a Web based system for keeping track of restaurant visits for a Diners' Society. The idea is that members of the Diners' Society can suggest a dinner at a restaurant, at a given date, and that other members of the society can register that they intend to attend the dinner. After the day of the dinner, members that have attended the dinner can rate the restaurant visit. The main page shows the average rating for each past restaurant visit together with an overview of future suggested visits.

## Exercise 1 (10 percent) - HTML

The following figure shows the login page of the Diners' Society Web-based system:

<b>Log into Diners' Society</b>		
Please enter email and password to log into Diners' Society:		
Email:	<input type="text" value="frede@fredensborg.dk"/>	
Password:	<input type="password" value="*****"/>	
	<input type="button" value="Log in"/>	
<a href="#">Diners' Society home</a>		<a href="mailto:webmaster@DinersSociety.com">webmaster@DinersSociety.com</a>

The login page (implemented as a file `login.php`) shows a form in which a member can enter an email address and a password. When the "Log in" button is pressed, the browser should send a request to the script `dologin.php` located at the URL `http://www.DinersSociety.com`. The names associated with the two text fields, for entering an email address and a password, should be `email` and `passwd`, respectively. The form should be presented in an HTML table with two columns and three rows.

The title of the login page should appear in the HTML code both in a `title` element (inside the `head` element) and as a heading (`h1`) in the body of the HTML code. The image appearing in the right part of the page can be assumed to be located in a file `champagne.gif` on the Web server.

At the bottom of the login page, two links are shown:

- a link to the login page itself with the text "Diners' Society home"
- a mailto-link to the email-address `webmaster@DinersSociety.com` with the text "webmaster@DinersSociety.com"

### Exercise 1.1 (10 percent)

Write the HTML code for the login page. (Recall that a file containing HTML code is a valid PHP script.)

### Exercise 2 (15 percent) - PHP

A site map diagram for the Diners' Society Web-based system is shown in Appendix A.

The boxes in the diagram represent Web pages (implemented as PHP scripts), the arrows without annotations represent HTML links, and the annotated arrows represent transactions (implemented as PHP scripts that use the HTTP redirect mechanism). The "Log out" links in the diagram implicitly redirect the user to the login page.

#### Exercise 2.1 (7 percent)

Construct a PHP function `pagehead` for writing beginning HTML code to the browser. The function should take two arguments:

1. a title to show in the title heading (`title`) and in the body heading (`h1`)
2. a member name to show in a *login status line* (e.g., "Frede is logged in. Log out")

If the member name given to the function is the empty string (""), no login status line should be displayed. This feature can be used for implementing the script `login.php`, which does not show a login status line.

The function should return a `html` begin-tag to the browser together with a `head` element containing the title of the page (given as argument to the function). The function should also return a `body` begin-tag to the browser together with an image link (as in Exercise 1.1), a heading (`h1`), and possibly, a login status line containing the user's member name and a link to the login page.

**Notice:** After a PHP script has called the function `pagehead`, the script can use the built-in PHP function `echo` for returning more HTML code to the user's browser.

#### Exercise 2.2 (8 percent)

Construct a PHP function `pagefoot` for writing the closing HTML code to the browser. The function should take two arguments:

1. a member id (`mid`), which identifies a member of the system
2. a password (`passwd`)

The function should return a horizontal ruler to the browser (`<hr/>`) and two links presented in an HTML table with two fields, one that is left aligned and one that is right aligned.

If both arguments to the function contain the empty strings (""), the first link should target the `login.php` script. Otherwise, the first link should target the script `index.php` with two form

variables `mid` and `passwd` containing the member id and the password, respectively (both given as arguments to the function).

The second link should be a mailto-link as specified in Exercise 1.1.

Finally, the function should return, to the browser, HTML code that closes the `body` and `html` tags.

### **Exercise 3 (15 percent) - Regular expressions**

#### **Exercise 3.1 (5 percent)**

A *well-formed password* consists of at least three characters, each of which can be either a lower case letter or a digit.

Construct a regular expression that is matchable only by well-formed passwords.

#### **Exercise 3.2 (5 percent)**

Construct a PHP function `validate_password` for checking whether a password is well-formed, as described in Exercise 3.1. If the argument passed to the function is a well-formed password, the function should return immediately. Otherwise, the function should display an error message in the user's browser and stop the program by calling the built-in PHP function `exit`.

Use the PHP function `ereg` to check that the password is well-formed.

#### **Exercise 3.3 (5 percent)**

A *well-formed time* consists of two parts separated by a colon (:). The first part should be a number between 0 and 23 (both numbers inclusive). The second part should be a number between 0 and 59 (both numbers inclusive). For the first part, a prefixed zero (0) is optional if the number is between 0 and 9 (both numbers inclusive). For the second part, a prefixed zero (0) is mandatory if the number is between 0 and 9 (both numbers inclusive).

Examples of well-formed times are 23:59, 5:09, and 03:00. Examples of strings that are not well-formed times are 23:2 and 24:00.

Construct a regular expression that is matchable only by well-formed times.

### **Exercise 4 (25 percent) - SQL**

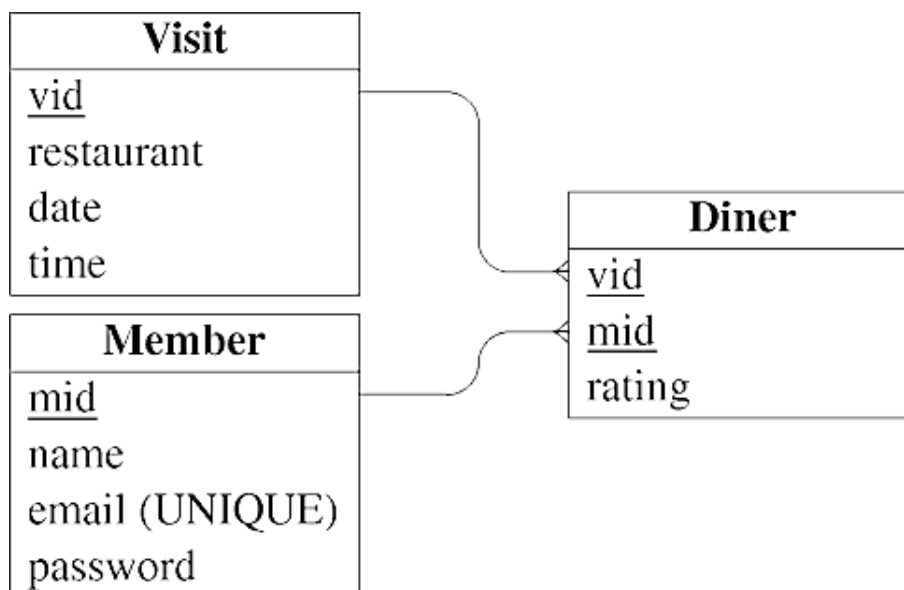
The data model for the Diners' Society system consists of three tables:

1. `visit` - contains data concerning restaurant visits
2. `member` - contains data concerning members of the society
3. `diner` - contains data relating members to visits

To simplify the system, the data model does not support multiple societies. Moreover, restaurants are referenced only by name; two visits to the same restaurant are modelled by two distinct rows in the `visit` table.

The data model is shown below in an E/R-diagram. The boxes represent the entities (tables), while

one-to-many relationships are indicated by forks. The upper half of each box contains the name of the entity, while attributes are listed in the lower part of the box. The primary key of an entity is composed by the underlined attributes in an entity.



The tables `Visit` and `Member` are created in MySQL using the following SQL statements:

```

CREATE TABLE Visit (
  vid      int PRIMARY KEY AUTO_INCREMENT,
  restaurant varchar (30) NOT NULL,
  date     date NOT NULL,
  time     time NOT NULL
) type = InnoDB;

CREATE TABLE Member (
  mid      int PRIMARY KEY AUTO_INCREMENT,
  name     varchar (50) NOT NULL,
  email    varchar (50) NOT NULL UNIQUE,
  password varchar (20) NOT NULL
) type = InnoDB;
  
```

In the above SQL statements, the built-in MySQL `auto_increment` feature is used to create unique identification numbers for each row in the tables.

You can assume that the following SQL statements have been executed:

```

-- Transactions for inserting new members
INSERT INTO Member (mid, name, email, password)
  VALUES (1, 'Margrethe', 'mor@amalienborg.dk', 'cigaret');
INSERT INTO Member (mid, name, email, password)
  VALUES (2, 'Frede', 'frede@fredensborg.dk', 'snorkel');
INSERT INTO Member (mid, name, email, password)
  VALUES (3, 'Henrik', 'henrik@amalienborg.dk', 'vin');
INSERT INTO Member (mid, name, email, password)
  VALUES (4, 'Mary', 'mary@fredensborg.dk', 'hophophop');
INSERT INTO Member (mid, name, email, password)
  VALUES (5, 'Joachim', 'joachim@schackenborg.dk', 'traktor');

-- Transactions for inserting new visits
INSERT INTO Visit (vid, restaurant, date, time)
  VALUES (1, 'De Gaulle', '2004-05-19', '19:00');
  
```

```
INSERT INTO Visit (vid, restaurant, date, time)
VALUES (2, 'Le Basilic', '2004-05-27', '19:30');
INSERT INTO Visit (vid, restaurant, date, time)
VALUES (3, 'Era Ora', '2004-06-09', '19:00');
```

#### Exercise 4.1 (4 percent)

Write down the result of executing the following `SELECT` statement - today is June 8, 2004:

```
SELECT date, time, restaurant
FROM Visit
WHERE date < now()
ORDER BY date, time;
```

#### Exercise 4.2 (4 percent)

Write down the SQL statement for creating the table `Diner`. The table should have three columns:

- `vid` - a reference to the `Visit` table
- `mid` - a reference to the `Member` table
- `rating` - an integer rating between 0 and 5 (both inclusive); `NULL` if the visit has not yet been rated by the member

Use the `NOT NULL` constraint to express the fact that the fields `vid` and `mid` must contain some data. Moreover, use the `PRIMARY KEY` constraint to express that the `vid` and `mid` columns together form the primary key of the table.

#### Exercise 4.3 (4 percent)

Write three SQL `INSERT` statements for inserting three diner ratings in the `Diner` table for the visit to the restaurant "De Gaulle 2004-05-19 at 19:00". The ratings should be for the members Henrik, Joachim, and Mary, who have rated as follows:

- Henrik: 5
- Joachim: 3
- Mary: 3

#### Exercise 4.4 (4 percent)

Construct an SQL `UPDATE` statement for updating the column `rating` (to contain the integer 5) in the table `Diner` for the association of the person Frede to the restaurant visit "De Gaulle 2004-05-19 at 19:00".

#### Exercise 4.5 (4 percent)

Construct an SQL `SELECT` query that shows future visits registered by some member of the society. The query should show the visit id (`vid`), the `date`, the `time`, and the restaurant name of the visit together with the number of members that have registered for the visit.

**Notice:** In order to show entries for which no member have registered, an inner join is necessary. You also need to use the SQL `GROUP BY` construct.

### Exercise 4.6 (5 percent)

Construct an SQL `SELECT` query that shows past visits registered by some member of the society. The query should show the visit id (`vid`), the date, the time, and the restaurant name of the visit together with the number of members that have registered for the visit and the average rating for those members that have rated the visit.

### Exercise 5 (35 percent) - Web-service

It can be assumed that the library `dslib.php` includes the `mydb.php` library from the lectures. It can also be assumed that the library `dslib.php` contains the function `validate_password` from Exercise 3.2 and functions `validate_int`, `validate_email`, `validate_rating`, `validate_date`, `validate_time`, and `validate_name` for validating integers, email addresses, ratings (i.e., integers between 0 and 5), dates, times, and restaurant names, respectively.

### Exercise 5.1 (7 percent)

Construct a function `checkpasswd` that takes two arguments, a member id (`mid`) and a password (`passwd`). The function should check whether the argument password is identical to the password in the `Member` table for the given member. If the passwords are not identical, the function should redirect the user to the login page and exit the script using the built-in PHP function `exit`. The function can assume that a connection to the database has already been established.

The function could possibly take the following form:

```
function checkpasswd ($mid, $passwd) {
    // Check that $mid and $passwd are well-formed
    ...

    // Query returning a row only if passwd is identical
    // to the password in the database
    $query = " ... ";
    $rows = mysql_query ($query);
    if (mysql_fetch_row ($rows))
        return;

    // Redirect the browser to the login page and exit
    ...
}
```

**Notice:** The function should use a call to the PHP function `header` to redirect the user to the `index.php` page.

### Exercise 5.2 (7 percent)

In this exercise, you are asked to implement part of the `index.php` script. Here is a template for the script:

```
<?php
include ("dslib.php");
mydb_connect ();
checkpasswd ($mid, $passwd);
$name = getmembername ($mid, $passwd);
pagehead ("Diners' Society", $name);
```

```

$stablestart =
  "<table cellpadding=3 cellspacing=2 width=80% bgcolor=lightgrey>
    <td><b>Date</b></td><td><b>Time</b></td>
      <td width=40%><b>Restaurant</b></td>
      <td><b>Diners</b></td>";

echo "<h3>Future visits:</h3>
<form action=addvisit.php>
  <input type=hidden name=mid value=$mid>
  <input type=hidden name=passwd value=$passwd>
  $stablestart <td><b>Action</b></td>";

$futureQuery = " -- QUERY FROM EXERCISE 4.5 --";
$rows = mysql_query ($futureQuery);
while ($row = mysql_fetch_row ($rows)) {
  $vid      = $row [0];
  $date     = $row [1];
  $time     = $row [2];
  $restaurant = $row [3];
  $diners   = $row [4];
  echo "<tr>
      <td>$date</td><td>$time</td><td>$restaurant</td>
      <td align=center>$diners</td>
      <td><a href='attend.php?vid=$vid&mid=$mid&passwd=$passwd'>Attend</a>
    </td></tr>";
}

echo "<tr><td><input type=text name=date size=10></td>
  <td><input type=text name=time size=5></td>
  <td><input type=text name=rest size=20></td>
  <td align=center>0</td>
  <td><input type=submit value='Add Visit'></td></tr>
</table></form>
<p>Enter dates in the format YYYY-MM-DD and times
  in the format HH-MM (0-24 hours).</p>";

$pastQuery = " -- QUERY FROM EXERCISE 4.6 --";
$rows = mysql_query ($pastQuery);
if ($row = mysql_fetch_row ($rows)) {

  // Echo Past visits header (h3) and start of table, including
  // table headers

  ##A##
}
while ($row) {
  $vid      = $row [0];
  $date     = $row [1];
  $time     = $row [2];
  $restaurant = $row [3];
  $diners   = $row [4];
  $rating   = $row [5];

  // Echo data for a past visit; include a link (Rate) to the
  // rate.php script, which takes as form variables a visit id (vid),
  // a member id (mid), and a password (passwd)

  ##B##

  $row = mysql_fetch_row ($rows);
}

echo "</table>\n";
pagefoot ($mid, $passwd);
?>

```

Write the PHP code for the program points `##A##` and `##B##` in the template above.

**Notice:** The function `getmembername`, which can be assumed to be defined in the `dslib.php` library, takes a member id (`mid`) as argument and returns the name of the member, as it appears in the `Member` table.

### Exercise 5.3 (7 percent)

Write the names of the form variables that the script `addvisit.php` receives when the form in the `index.php` script is submitted (see the code for the `index.php` script in Exercise 5.2).

Construct the file `addvisit.php`. Remember to check the validity of the form variables and that the user is authenticated (i.e., that the password is correct).

**Notice:** The script should redirect the user to the `index.php` script, which requires the form variables `mid` and `passwd`.

### Exercise 5.4 (7 percent)

A template for the script `rate.php` is given as follows:

```
<?php
include ("dslib.php");
mydb_connect ();
checkpasswd ($mid, $passwd);
$name = getmembername ($mid, $passwd);
pagehead ("Diners' Society: Rate a visit", $name);
validate_int ($vid);

$rows = mysql_query ("SELECT date, time, restaurant
                     FROM Visit WHERE Visit.vid = $vid");

if ($row = mysql_fetch_row ($rows)) {
    $date = $row [0];
    $time = $row [1];
    $restaurant = $row [2];
} else {
    error ("Something went wrong: unknown visit id");
}

echo "<p>The following ratings exist for $restaurant
      $date at $time:</p>
      <form action=addrating.php><table>";

$rows = mysql_query
("SELECT Diner.mid, name, rating
 FROM Member, Diner
 WHERE Member.mid = Diner.mid AND Diner.vid = '$vid'
 ORDER BY name");
##C##

echo "</table>
      <input type=hidden name=vid value=$vid>
      <input type=hidden name=mid value=$mid>
      <input type=hidden name=passwd value=$passwd>
      </form>";

pagefoot($mid, $passwd);
?>
```

Write the code at program point `##C##` for showing restaurant visit ratings for each of the visiting members. For the visiting member currently logged in, a form should be displayed instead of just the rating as text. The form should target the script `addrating.php` and contain a button labelled "Rate" and a text field with name `rating` and the value equal to the previous rating (if not NULL).

### **Exercise 5.5 (7 percent)**

Construct the script `addrating.php`. The script should update the rating, as it is entered by the member in form, and redirect the user to the script `index.php`.

# Appendix A: Site Map Diagram for the System

login.php

### Log into Diners' Society

Please enter email and password to log into Diners' Society:

Email:

Password:



[Diners' Society home](#) [webmaster@DinersSociety.com](mailto:webmaster@DinersSociety.com)

dologin.php

index.php

### Diners' Society

Frede is logged in [Log out](#)

Future visits:

Date	Time	Restaurant	Diners	Action
2004-05-09	19:00	Era Ora	2	<a href="#">Attend</a>
<input type="text" value="2004-07-25"/>	<input type="text" value="18:00"/>	<input type="text" value="Restaurationen"/>	0	<input type="button" value="Add Visit"/>

Enter dates in the format YYYY-MM-DD and times in the format HH:MM (0-24 hours)

Past visits:

Date	Time	Restaurant	Diners	Rating	Action
2004-05-19	19:00	De Gaulle	4	3.75	<a href="#">Rate</a>
2004-05-27	19:00	Le Basilic	2	3.00	<a href="#">Rate</a>

[Diners' Society home](#) [webmaster@DinersSociety.com](mailto:webmaster@DinersSociety.com)

addvisit.php

cancel.php

addrating.php

### Diners' Society: Attend a visit

Frede is logged in, [Log out](#)

Members currently signed up for visiting Era Ora 2004-06-09 at 19:00:00:

- Frede [cancel](#)
- Joachim
- Margrethe

[Diners' Society home](#) [webmaster@DinersSociety.com](mailto:webmaster@DinersSociety.com)

attend.php

### Diners' Society: Rate a visit

Frede is logged in, [Log out](#)

The following ratings exist for De Gaulle 2004-05-19 at 19:00:00:

Frede:

Henrik: 5

Joachim: 3

Mary: 3

[Diners' Society home](#) [webmaster@DinersSociety.com](mailto:webmaster@DinersSociety.com)

rate.php