

COMP3322A Modern Technologies on World Wide Web

Project I – Online cinema ticket ordering and movie reviewing

Total 28 points

Due Date: 5pm November 6, 2018

Overview:

In this assignment, you are going to design and implement an online cinema ticket ordering, movie reviewing and purchase history reviewing system that supports registered users to:

- buy cinema ticket(s) according to their own preferences and seat availability;
- give comments and view other users' comments on the selected film(s);
- view their own ticketing history.

In the assignment, you will use different techniques and technologies, which include HTML, CSS, JavaScript, MySQL, AJAX, and PHP, to construct and implement different web pages for collecting users' input from the client side and to retrieve data from the server side that corresponds to users' input. You don't have to design your own database schema as we shall provide it to you (but you can also design your own if you are interested in it). However, your system must fulfill the functionalities and layout requirement as stated in this assignment document (in the specification part).

Objectives:

1. A learning activity to support ILO 2.
2. The goals of this programming assignment are:
 - to get solid experience in using client side, server-side techniques, and database system to design and implement a web-based application which contains some common features of an online cinema ticketing and reviewing system.
 - to get a good understanding of how the layout of a web-based application can be enhanced by implementing CSS and responsive web design.

Specifications:

This assignment consists of three parts:

- **Part I**
Implement the basic skeleton of the login page and create account page by using HTML and basic JavaScript (no decoration of web page by using CSS is needed).
- **Part II**
Implement the functionalities of the system, which mainly consists of 3 components/services: Buy ticket(s), movie review and purchase history. Majority of functions/services are created by using the server-side techniques: PHP, MySQL, use of session and AJAX.
- **Part III**
Design the layout of the web pages by using CSS and responsive web design. In this part, you should use CSS to decorate all web pages. In addition, you are required to apply responsive web design to the series of the web pages that are related to "Buy a ticket".

In summary, you are going to develop the system incrementally. First, go through Part 1 to design a simple login system, then enhance it to Part 2 with more functionalities. Finally, you would enhance the layout and appearance of the system after going through part 3.

Part 1 (Total: 4 points)

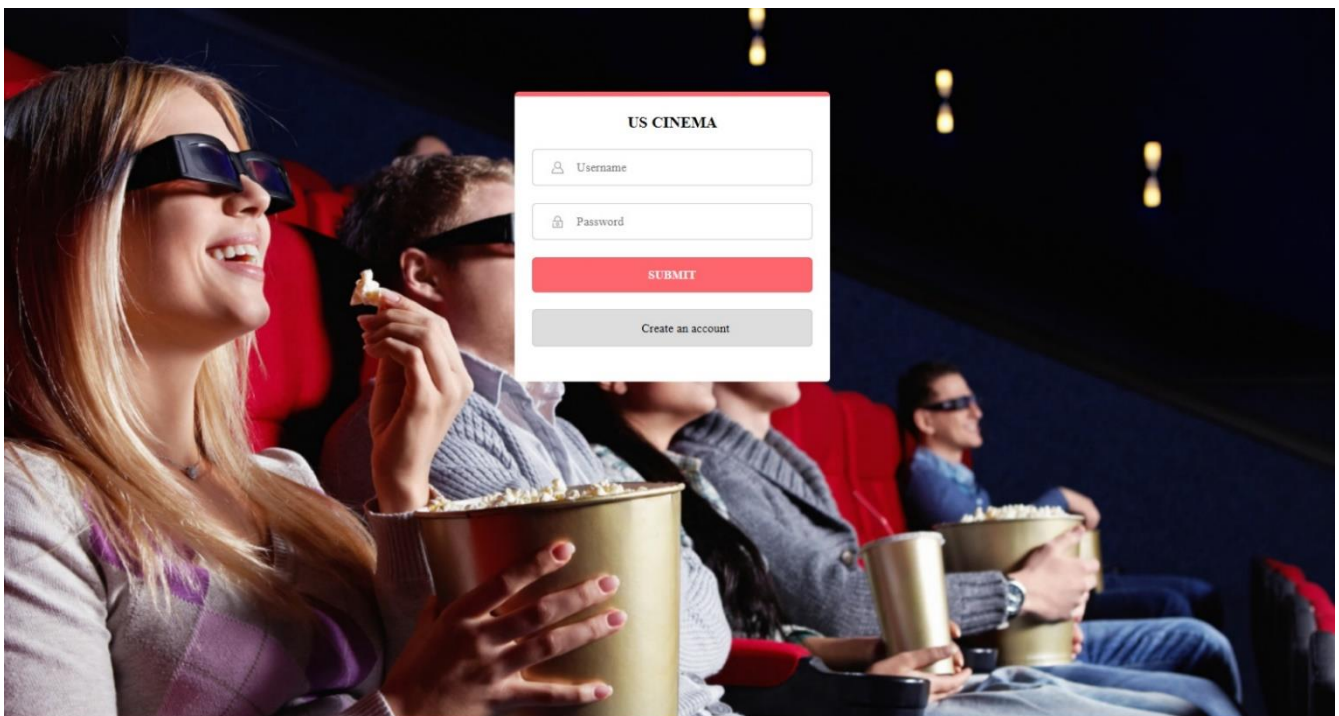
You are expected to use HTML and JavaScript to construct the login system which comprises two HTML pages: ***index.html*** and ***createaccount.html***.

index.html – it provides the following items and services (2 points):

- [Login] form – A login form which uses the method of “POST” to send the data that is entered by the user in the username and password textboxes. It is expected that the data will be passed to a php file called “***verifyLogin.php***” which will be implemented in Part 2. Besides, before submitting the form to the server, you should use JavaScript ***or other means*** to check whether the textbox(es) is/are empty; if yes, display an alert showing "Please do not leave the fields empty" to the user. You should also include the name of the cinema at the top of the page with HTML heading <h1> (The name is designed by you).
- [Username] textbox- A textbox which allows the user to enter his/her account name with a placeholder “Username”.
- [Password] textbox- A textbox which allows the user to enter his/her account password with a placeholder “Password”. The password should not be visible to others when the user is entering in the textbox.
- [Submit] button- A submit button which the form data (the username and password) will be sent to the php file “***verifyLogin.php***” (this file will be implemented in Part 2 later) once it is clicked. The button should have a display name “SUBMIT”.
- [Create account] button- A button which when clicked, it will direct to the page “***createaccount.html***”. The button should have a display name “Create an account”.

Figure 1 shows the sample of “index.html”. Note that in Figure 1, CSS code is included to decorate the page, but you DO NOT need to implement CSS in this stage, implementing the basic skeleton as stated above is fine.

Figure 1



createaccount.html – it provides the following items and services (2 points):

- [Create] form- A create account form which uses the “POST” method to send the data that is entered by the user in the new username textbox and the new password textbox. It is expected that the data

will be passed to a php file called “**create.php**” which will be implemented in Part 2 later. Besides, before submitting the form to the server, you should use JavaScript **or other means** to check the following conditions:

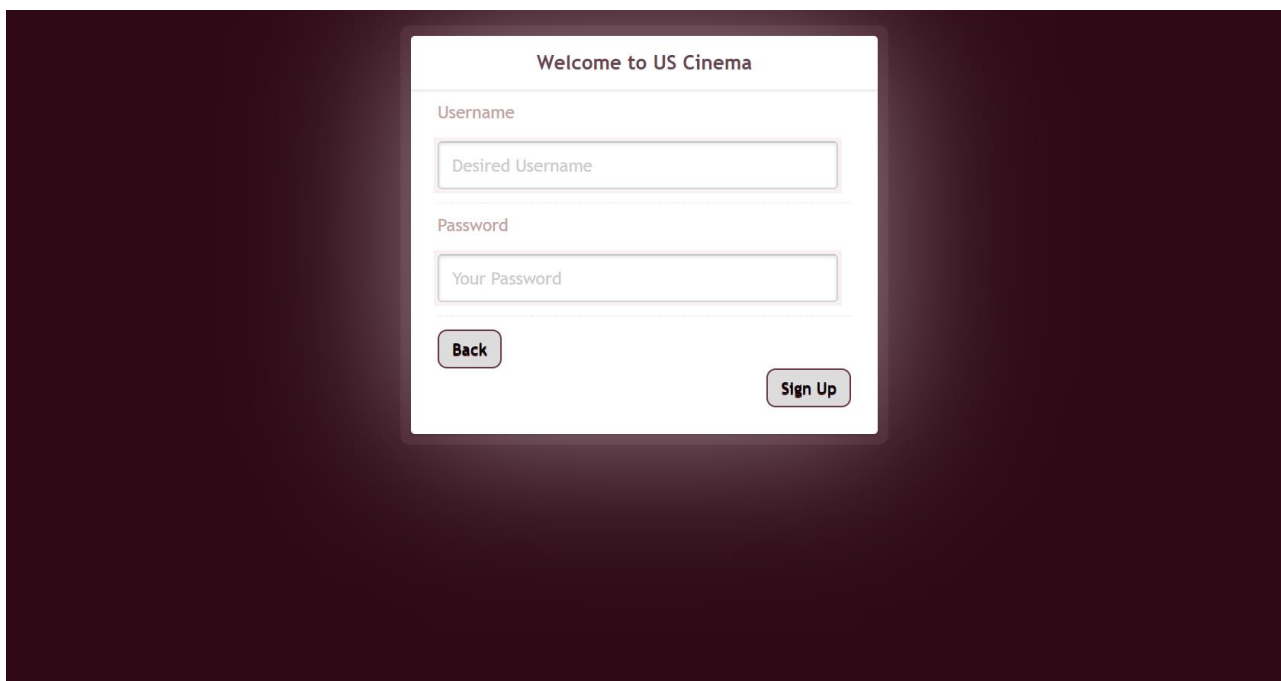
- Whether the username textbox and password textbox are empty. If yes, display an alert “Please do not leave the fields empty” to the user.
- Whether the length of the new username entered by the user is too short (i.e. fewer than 3 characters). If yes, display an alert “Username must be longer than 3 characters” to the user.
- Whether the new username entered by the user contains any special characters. If yes, display an alert “Username can only consist of characters and numbers” to the user.
- Whether the password is too short (less than 6 characters) or too long (longer than 15 characters) or not composed of only alphanumerical characters with at least one alphabet and one numerical character. If yes, display an alert “The length of the password must be between 6 to 15 characters and it must consist of only alphanumerical characters with at least one alphabet and one numerical character” to the user.

As the “**create.php**” file is not ready yet, the form data entered by the user should not be submitted to the server at this stage.

- [Username] textbox- A textbox which allows the user to enter his/her new account name with a placeholder “Desired Username”.
- [Password] textbox- A textbox which allows the user to enter his/her new account password with a placeholder “Your Password”. The password should not be visible to others when the user is entering in the textbox.
- [Back] button- A button which when clicked, it will direct back to the page “index.html”. The button should have a display name “Back”.
- [Sign up] button- A submit button which the form data (the new username and new password) will be sent to the php file “**create.php**” (this file will be implemented in Part 2 later) once it is clicked. The button should have a display name “Sign Up”.

Figure 2 illustrates the sample of “createaccount.html”. Note that in Figure 2, CSS code is included to decorate the page, but you DO NOT need to implement CSS in this stage, implementing the basic skeleton as stated above is fine.

Figure 2



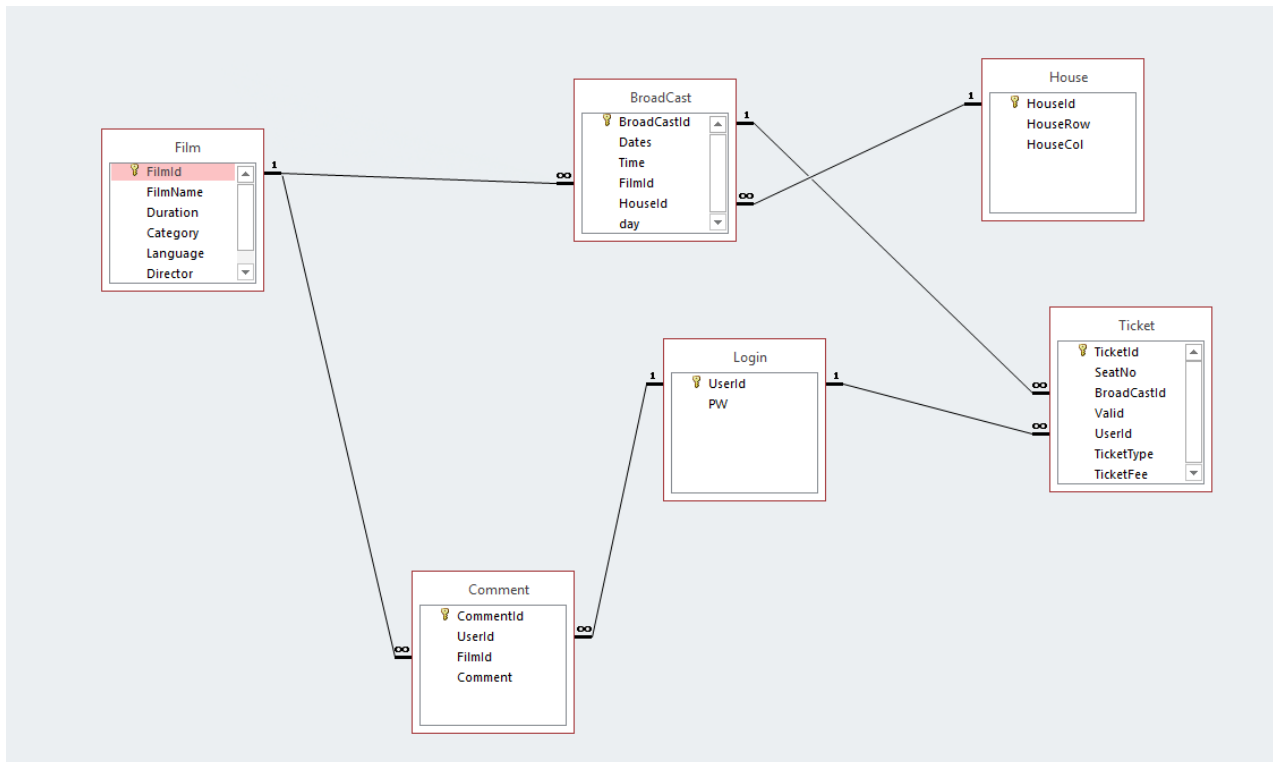
The image shows a web form titled "Welcome to US Cinema" for creating a new account. The form is centered on a dark purple background. It contains two input fields: "Username" with a placeholder "Desired Username" and "Password" with a placeholder "Your Password". Below the fields are two buttons: "Back" and "Sign Up".

Part 2 (Total: 12 points)

For this part, you are going to implement the server-side scripting by using PHP, MySQL, session and AJAX to receive requests from the client side; based upon the request, retrieve data from the MySQL database and deliver suitable response back to the client. There are in totally **12 php files** that you have to implement in this part.

However, before creating the php files, you should create a database schema in the MySQL server first. The suggested database schema is as follow (in the next page) (you can follow it, or you can create your own database schema if you want to, as long as you can fulfill the specification mentioned in this part).

Figure 3 Suggested database scheme



[Film table] – It consists of the following fields:

- FilmId (Primary key)
- FilmName
- Duration
- Category
- Language
- Director
- Description

[Comment table] – It consists of the following fields:

- CommentId (Primary key)
- FilmId (Foreign key, reference to the Film table)
- UserId (Foreign key, reference to the Login table)
- Comment

[BroadCast table] – It consists of the following fields:

- BroadCastId (Primary key)
- Dates
- Time
- FilmId (Foreign key, reference to the Film table)
- HouseId (Foreign key, reference to the House table)
- day

Note: day means the film broadcast day, i.e. from Monday to Sunday.

[Ticket table] – It consists of the following fields:

- TicketId (Primary key)
- SeatNo
- BroadCastId (Foreign key, reference to the BroadCast table)
- Valid
- UserId (Foreign key, reference to the Login table)
- TicketType
- TicketFee

Note:

SeatNo means seat number, if the current seat is located at row A, column 1, then the seat number is A1.

Valid means availability of the seat, i.e. whether it is sold or not.

[House table] – It consists of the following fields:

- HouseId (primary key)
- HouseRow
- HouseCol

Note:

HouseRow means the last seat row in the house.

HouseCol means the last seat column in the house.

[Login table] – It consists of the following fields:

- UserId (Primary key)
- PW

Note: PW means the login password.

The first 2 php files are “**verifyLogin.php**” and “**create.php**”. The former script is used to verify whether the username and password entered by the user in “index.html” match with the username and password stored in the MySQL database. The latter script is used to check whether the username entered by the user in “createaccount.html” already exists in the MySQL database.

verifyLogin.php – it provides the following items and services (1 point):

- Receive the username and password passed from the login form and check whether the users’ input matches with the username and password stored in the Login table. If yes, create a session variable to store the username (as an indication that the session is active) and then redirect the user to “**main.php**” (which will be implemented later). If no, display the message “Invalid login, please login again.” (with <h1> tag) and then redirect back to “**index.html**” after 3 seconds.

create.php – it provides the following items and services (1 point):

- Receive the username and password passed from the createaccount form and check whether the username inputted by the user already exists in the Login table. If yes, display the message “Account already existed” (with <h1>tag) and then redirect back to “**createaccount.html**” page after 3 seconds. If no, insert the username and password into Login table as a new record, print the message “Account created! Welcome” (with <h1>tag) and redirect back to “**index.html**” after 3 seconds.

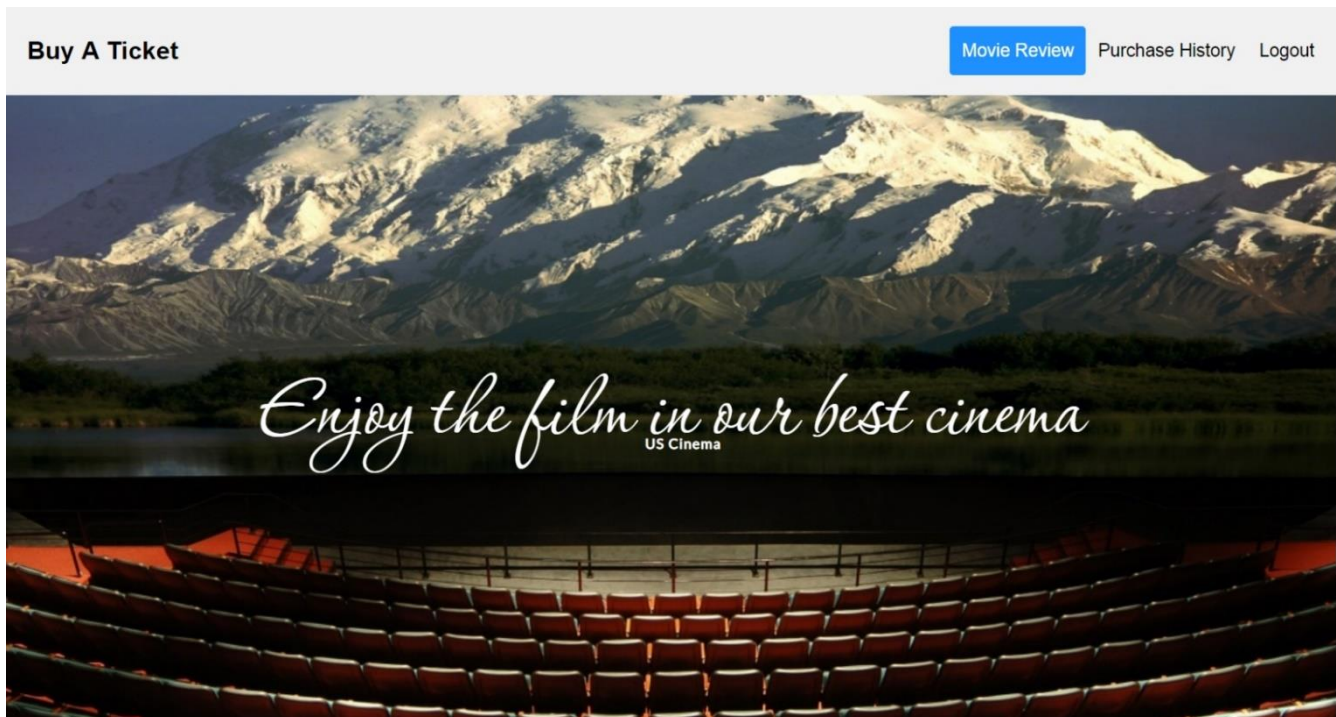
The php file “**main.php**” serves as the landing page after user successful logged on. It only consists of links lead to services (other php scripts) provided by this site.

main.php – it provides the following items and features (0.5 points):

- On the top of the page, there have 4 hyperlinks:
 - Buy A Ticket- This link should link to the “**buywelcome.php**” script (which will be implemented later).

- Movie Review- This link should link to the “**comment.php**” script (which will be implemented later).
- Purchase History- This link should link to the “**history.php**” (which will be implemented later).
- Logout- This link should link to the “**logout.php**” (which will be implemented later).
- Below the hyperlinks is the body of the page. It only consists of an image of the cinema and some descriptions about the cinema (you can put whatever image and descriptions you like).
- This page should also check whether some user directly visits this page without login (i.e., without going through the index.html page). If yes, display “You have not logged in” and redirect back to index.html (the login page) after 3 seconds. A sample of “main.php” is shown below (You are suggested to use a session variable to check whether the user has logged in already).

Figure 4



As seen from the main page (main.php), this system mainly consists of 3 services which are “buying a ticket online”, “conducting movie review”, as well as “viewing the user’s ticket purchase history”. The following table summarizes the sets of php script files that are responsible for the services:

Feature	Files requested
Buy A Ticket	“ buywelcome.php ”, “ seatplantry.php ”, “ buyticket.php ”, “ confirm.php ”
Movie Review	“ comment.php ”, “ comment_submit.php ”, “ comment_retrieve.php ”
Purchase History	“ history.php ”
Logout	“ logout.php ”

We suggest you first focus on the feature “Buy A Ticket”.

buywelcome.php – it provides the following items and functions (1 point):

- Same as “**main.php**” page, on the top of the page, it is expected that there are 4 hyperlinks that link to the “**buywelcome.php**” page (Buy A Ticket) (current page), “**comment.php**” page (Movie Review) (which will be implemented later), “**history.php**” page (Purchase History) (which will be implemented later) and “**logout.php**” page (Logout) (which will be implemented later) respectively.

- Like “**main.php**” page, this page should check whether the user directly visits this page without login. If yes, display “You have not logged in” and redirect back to index.html (the login page) after 3 seconds.
- The body of the page should display a film list which shows all the current movies that are now being broadcasted. For each film, you should show the film name (with <h1> tag), a film poster, the synopsis (description) of the film (with <h3> tag), the director of the film (with <h4> tag), the duration of the film (with <h4> tag), the category of the film (with <h4> tag) and the language of the film (with <h4> tag). You may use a horizontal line (<hr>) to separate the content of each film; however, during styling using CSS, you can remove the horizontal line when appropriate. You are given with the information of 4 movies and you have to insert the information into the MySQL database. The script should compose the file list by retrieving the information from the database. Here are the contents of the movies:

○ **Film 1:**

Film Name:	Return Of The Cuckoo
Synopsis (Description):	During the day of the handover of Macau in 1999, Man-Cho (Chi Lam Cheung), Kiki (Joe Chen) and a group of neighbors were celebrating with Aunty Q (Nancy Sit) for her birthday. Kwan-Ho migrates to US for...
Director:	Patrick Kong
Duration:	103 mins
Category:	IIA
Language:	Cantonese
Film poster:	movie001. jpg

○ **Film 2:**

Film Name:	Suffragette
Synopsis (Description):	The foot soldiers of the early feminist movement, women who were forced underground to pursue a dangerous game of cat and mouse with an increasingly brutal State...
Director:	Sarah Gavron
Duration:	106 mins
Category:	IIA
Language:	English
Film poster:	movie002.jpg

○ **Film 3:**

Film Name:	She Remembers, He Forgets
Synopsis (Description):	Unfulfilled at work and dissatisfied with her marital life, a middle-aged woman attends a high school reunion and finds a floodgate of flashbacks of her salad days open before her mind eyes...
Director:	Adam Wong
Duration:	110 mins
Category:	IIA
Language:	Cantonese
Film poster:	P3.png

○ **Film 4:**

Film Name:	Spectre
Synopsis (Description):	A cryptic message from the past sends James Bond on a rogue mission to Mexico City and eventually Rome, where he meets Lucia Sciarra (Monica Bellucci), the beautiful and forbidden widow of an infamous criminal...
Director:	Sam Mendes
Duration:	148 mins
Category:	IIB

Language:	English
Film poster:	movie004.jpg

- In addition to the above information, each film should contain a pull-down menu which shows the availability broadcast options and a submit button which allows users to indicate their desired film broadcast option and submit it to the system. The data should be sent to a php file called “**seatplantry.php**” (which will be implemented next) which allows users to select their seat(s) after selecting their desired broadcast option. The following section shows the broadcast information for each film:

- For the Film 1 “Return Of The Cuckoo”, the following broadcast option should be displayed:

Broadcast option:	16/11/2015 12:10 (Mon) House A 16/11/2015 13:10 (Mon) House C
-------------------	--

- For the Film 2 “Suffragette”, the following broadcast option should be displayed:

Broadcast option:	16/11/2015 12:50 (Mon) House A 16/11/2015 13:20 (Mon) House B
-------------------	--

- For the Film 3 “She Remembers, He Forgets”, the following broadcast option should be displayed:

Broadcast option:	16/11/2015 15:20 (Mon) House A
-------------------	--------------------------------

- For the Film 4 “Spectre”, the following broadcast option should be displayed:

Broadcast option:	16/11/2015 16:20 (Mon) House A
-------------------	--------------------------------

Figure 5 shows part of the “buywelcome.php” page:

Figure 5

Return Of The Cuckoo



Synopsis: During the day of the handover of Macau in 1999, Man-Cho (Chi Lam Cheung), Kiki (Joe Chen) and a group of neighbors were celebrating with Auntie Q (Nancy Sit) for her birthday. Kwan-Ho migrates to US for..

Director: Patrick Kong

Duration: 103 mins

Category: IIA

Language: Cantonese

16/11/2015 12:10 (Mon) House A ▼ Submit

Suffragette



Synopsis: The foot soldiers of the early feminist movement, women who were forced underground to pursue a dangerous game of cat and mouse with an increasingly brutal State...

Director: Sarah Gavron

Duration: 106 mins

Category: IIA

Language: English

16/11/2015 12:50 (Mon) House A ▼ Submit

seatplantry.php – it provides the following items and features (2.5 points):

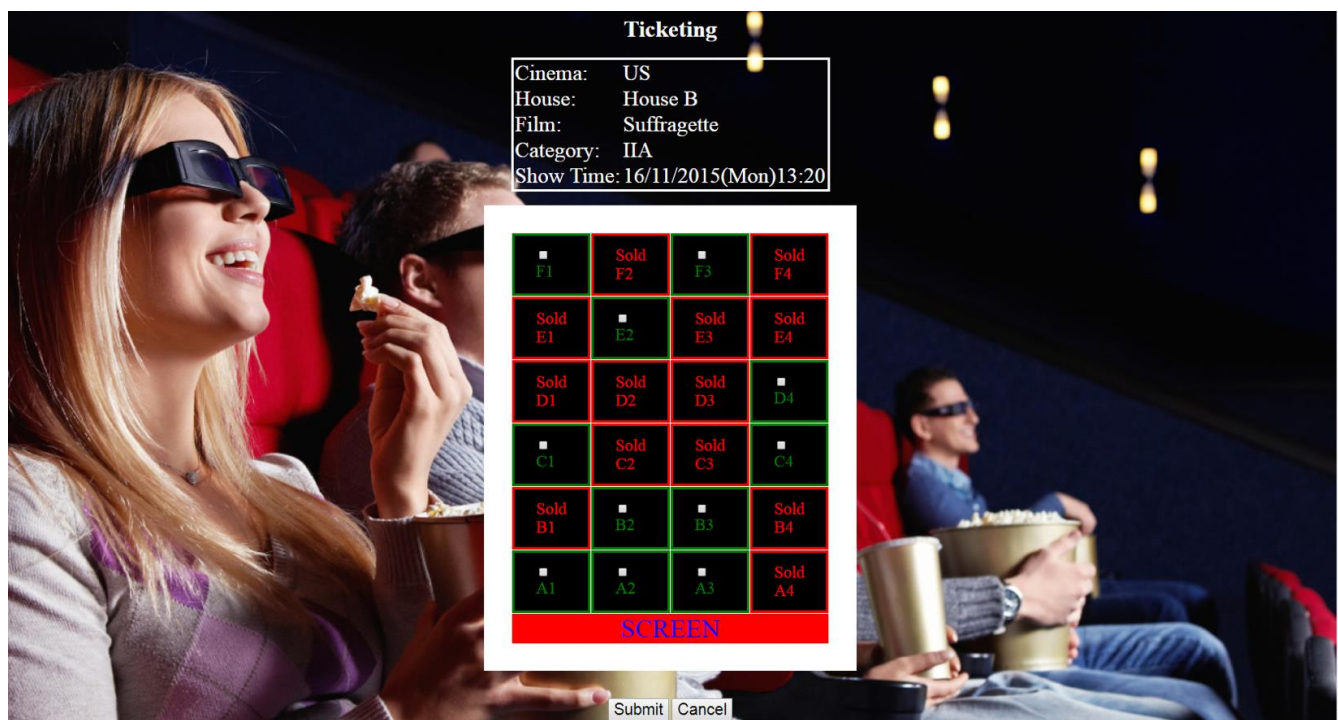
- This page should check whether the user directly visits this page without login. If yes, display “You have not logged in” and redirect back to index.html (the login page) after 3 seconds.
- There should have a header named “Ticketing” (with <h1> tag) at the top of the page.
- Based on the user’s selected broadcast option, below the header, there should have a table showing the broadcast information (the cinema name, the house name, the film name, the category and the broadcast time) of the film. Immediately follows this block is the seating plan of the corresponding house as shown in Figure 6. For a seat that is available, it should be displayed in green and there is a checkbox which allows the user to select. In contrast, for a seat that is not available, it would be expressed in red and has the word “Sold” without the checkbox. Before the first row of seats, there should have a row called “SCREEN” with a red background and blue bold font style and color. Note that the label of the seat is generated from the row and column indexes; for example, if you name the rows from A-Z and columns from 1 to 10, then the seat at the first row and the first column would be named as “A1”. It is also expected that once a seat is uniquely owned by a user in the system, it is considered as “Sold”, otherwise it should be available.
- In addition, under the seating plan, there should have two buttons named “Submit” and “Cancel”. The former button would submit the user’s seat selection to a page named “**buyticket.php**” (which will be implemented next) while the other button would direct back to “**buywelcome.php**” upon it is clicked. Please note that this page should check whether the user has selected any seat before moving to the “**buyticket.php**” page.

The information on the number of seats in different houses is as follow:

- House A: Total seats: 25, from row A to E (5 rows) and from 1 to 5 (5 columns).
- House B: Total seats: 24, from A to F (6 rows) and from 1 to 4 (4 columns).
- House C: Total seats: 28, from A to D (4 rows) and from 1 to 7 (7 columns).

The sample of the “**seatplantry.php**” page is in Figure 6.

Figure 6

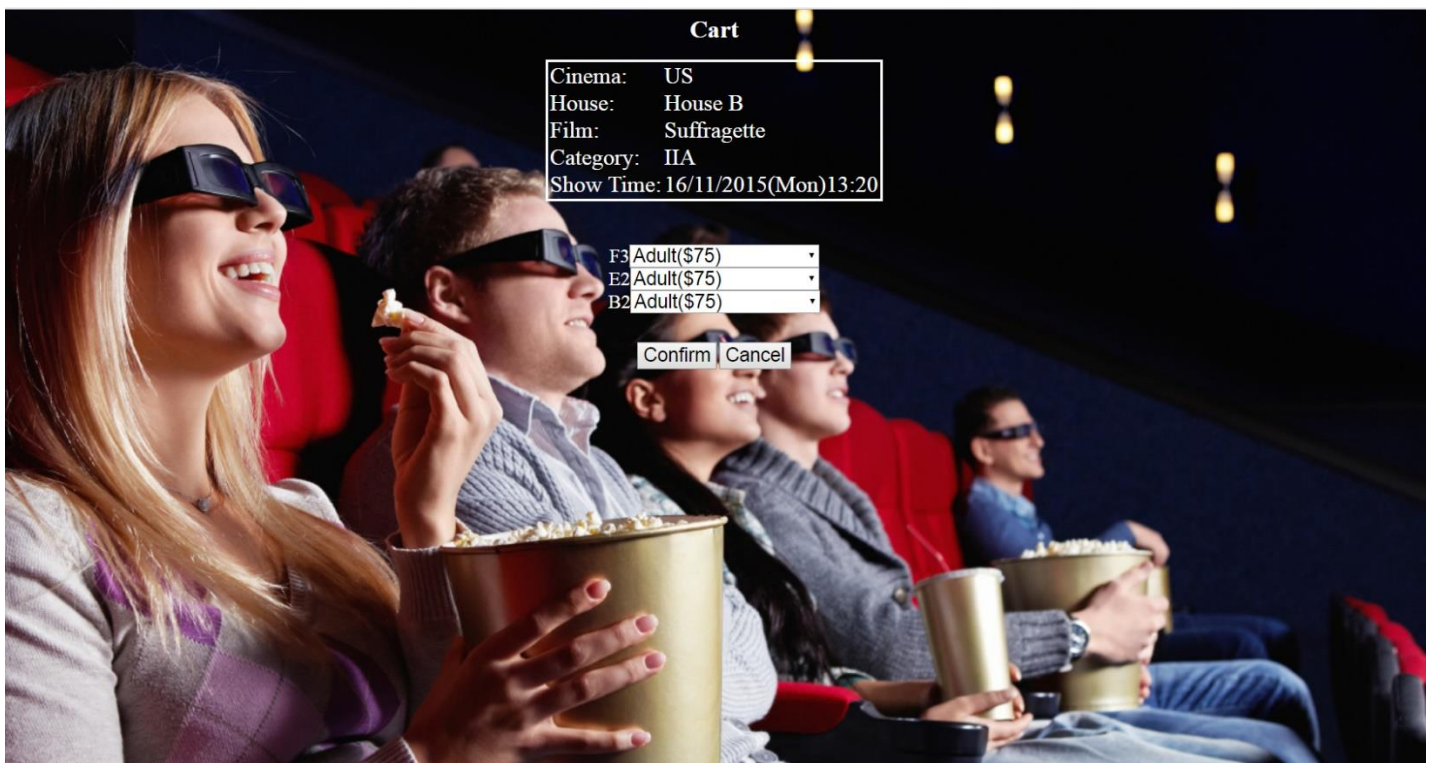


buyticket.php – it provides the following items and functions (1 point):

- This page should check whether the user directly visits this page without login. If yes, display “You have not logged in” and redirect back to index.html (the login page) after 3 seconds.
- The page should first have a heading (with <h1> tag) called “Cart”, followed by a table that shows the broadcast information (the cinema name, the house name, the film name, the category and the broadcast time of the film), which is the same as in “*seatplantry.php*”. Underneath the information table, it should display the seat(s) that the user has selected via “*seatplantry.php*”. Each seat should be presented as a row starting with the seat’s label and is followed by a pull-down menu with options “Adult(\$75)” or “Student/Senior(\$50)” for the user to select the type of ticket he/she wants to buy. The default is set to “Adult(\$75)”.
- A submit button named “Confirm” and a cancel button named “Cancel” is located underneath the selected seat(s). When the former button is clicked, the information selected by the user in this page would be sent to “*confirm.php*” (which will be implemented next), while if the latter button is clicked, it would direct the user back to the page “*buywelcome.php*”.

The sample of “*buyticket.php*” is shown in Figure 7.

Figure 7



confirm.php – it provides the following items and features (1 point):

- Like the “*main.php*” page, on the top of the page, there should be 4 hyperlinks that link to the page of “*buywelcome.php*” (Buy A Ticket), “*comment.php*” (Movie Review) (which will be implemented later), “*history.php*” (Purchase History) (which will be implemented later) and “*logout.php*” (Logout) (which will be implemented later).
- This page should check whether the user directly visits this page without login. If yes, display “You have not logged in” and redirect back to index.html (the login page) after 3 seconds.
- In the body of the page, there should be a heading (with <h1> tag) called “Order information”, followed by the order information of **each ticket** (the cinema name, the house name, the seat number, the film

name, the category of the film, the film show time and the ticket fee) that the user has confirmed in the “**buyticket.php**” page in the previous step. The total amount of payment that the user should pay is also displayed after the ticket information.

- At the bottom of the page, a reminder “Please present valid proof of age/status when purchasing Student or Senior tickets before entering the cinema house.” should be displayed, after a horizontal line break (<hr>). A button named “OK” should direct the page back to “**buywelcome.php**” page when clicked. You should note that when the user goes back to “**buywelcome.php**” page and select the same film with the same broadcast time, the seat(s) that he/she has just purchased should be displayed as “Sold”.

The sample of “confirm.php” is illustrated on Figure 8.

Figure 8

Buy A Ticket [Movie Review](#) [Purchase History](#) [Logout](#)

Order Information

Cinema:	US Cinema
House:	House A
SeatNo:	D3
Film:	Spectre
Category:	IIB
Show Time:	16/11/2015(Mon)16:20
Ticket Fee:	\$75(Adult)

Cinema:	US Cinema
House:	House A
SeatNo:	C3
Film:	Spectre
Category:	IIB
Show Time:	16/11/2015(Mon)16:20
Ticket Fee:	\$75(Adult)

Total fee: \$ 150

Please present valid proof of age/status when purchasing Student or Senior tickets before entering the cinema house.

OK

Next, we would implement the feature “Movie Review” by creating 3 php files “comment.php”, “comment_submit.php” and “comment_retrieve.php”. Like other pages, check whether the user directly visits these pages without login and take appropriate action if needed.

comment.php – it provides the following items and functions (2 points):

- On the top of the page, there should be 4 hyperlinks that link to the page of “**buywelcome.php**” (Buy A Ticket), “**comment.php**” (Movie Review) (current page), “**history.php**” (Purchase History) (which will be implemented later) and “**logout.php**” (Logout) (which will be implemented later).

- Like other pages, this page should check whether the user directly visits this page without login. If yes, display “You have not logged in” and redirect back to index.html (the login page) after 3 seconds.
- In the body of the page, there should be a pull-down menu which allows the user to select the film that he/she wants to make comment. The film name should be retrieved from MySQL database. Underneath the film name, there should be a text area which allows the user to enter a comment. The text area should have a row size of 20, a column size of 80, with a placeholder “Please input comment here”.
- Below the comment text area, there should have two buttons which are the “View comment” and “Submit comment” buttons. When the user clicks on the “View comment” button, it is expected that an **AJAX** GET request (containing the film name) should be sent to “**comment_retrieve.php**” page (which will be implemented next) for the retrieval of comments from the MySQL database. The retrieval process should be handled by “**comment_retrieve.php**”.
- When the “Submit comment” button is clicked, it is expected that a POST request carrying the information of the film name and the user’s comment is sent to “**comment_submit.php**” page for further processing (which will be implemented next). Before submitting the POST request, the system should check whether the user has entered something in the text area. If the text area is empty, the system should alert the user to enter the comment before the submission.

Please view Figure 9 for reference.

Figure 9 Sample of “comment.php” when loaded

comment_retrieve.php – it provides the following functions (0.5 points):

- Upon receiving the **AJAX** request sent from “**comment.php**”, it should retrieve all the comments (and the corresponding viewers) on a particular film (which is selected by the user in “**comment.php**”) from the MySQL database. They should be displayed underneath the two buttons (“View comment” and “Submit comment”) with the name of the viewer (i.e. Viewer: (viewer name)) (<h2> tag) and the comment they have been submitted (<h3> tag). Every submission of comment should be separated by a horizontal line when displayed.

Please view Figure 10 for reference.

Figure 10 Sample of "comment_php" after "View comment" button is clicked

The screenshot shows a web application interface. At the top, there is a navigation bar with four links: "Buy A Ticket", "Movie Review" (highlighted in blue), "Purchase History", and "Logout". Below the navigation bar, there is a form for adding a comment. The form has a label "Film Name:" followed by a dropdown menu showing "Return Of The Cuckoo". Below this is a large text area with the placeholder text "Please input comment here". At the bottom of the form, there are two buttons: "View comment" and "Submit comment". Below the form, there is a list of comments. The first comment is by "Viewer: Marco" with the text "Comment: A great movie!". The second comment is by "Viewer: Mary" with the text "Comment: There is a saying that lightning never strikes the same place twice. Many of us, however, probably experienced a series of misfortunes at least once.".

Buy A Ticket Movie Review Purchase History Logout

Film Name: Return Of The Cuckoo ▼

Please input comment here

View comment Submit comment

Viewer: Marco
Comment: A great movie!

Viewer: Mary
Comment: There is a saying that lightning never strikes the same place twice. Many of us, however, probably experienced a series of misfortunes at least once.

comment_submit.php – it provides the following items and functions (0.5 points):

- Upon receiving the POST request, the submitted comment should be inserted into the MySQL database. The database should store the identity of the writer who wrote the comment, the film that the writer commented on, as well as the comment only if the comment content is NOT left empty. Upon successful insertion of the new comment record, the page should display a message "Your comment has been submitted" (with <h1> tag) and redirect the user back to "**comment.php**" page after 3 seconds.

Please view Figure 11 for reference.

Figure 11 Sample of "comment_submit.php" after the user successfully submitted the comment

Your comment has been submitted.

The next step is to implement the feature of "Purchase History" by creating a php file "**history.php**". Like other pages, check whether the user directly visits this page without login and take appropriate action if needed.

history.php – it provides the following items and functions (0.5 points):

- Same as "**main.php**", on the top of the page, there should be 4 hyperlinks to the page of "**buywelcome.php**" (Buy A Ticket), "**comment.php**" (Movie Review), "**history.php**" (current page) and "**logout.php**" (Logout) (which will be implemented later).
- In the body of the page, there should be a title named "Purchase History" (with <h1> tag), followed by displaying the username of the current user (with <h3> tag). The purchase history would be displayed in a format of **one record per each ticket that the user had purchased**. For example, the user purchased three tickets in one transaction, there would have 3 records in the "Purchase History". For each ticket, the ticket id, ticket fee, selected house, seat, film name, language and broadcast date should be displayed. A horizontal line should be inserted (<hr>) between each ticket record. [Note: you are allowed to remove the horizontal lines once styling is applied.]

Please view Figure 12 for reference.

Figure 12

Buy A Ticket	Movie Review	Purchase History	Logout
Purchase History			
Username: Mary			
TicketId:3001 \$75(Adult) House: A Seat: A2 FilmName: Return Of The Cuckoo(IIA) 103 mins Language: Cantonese Date: 16/11/2015(Mon) 12:10			
TicketId:3055 \$50(Student/Senior) House: B Seat: E4 FilmName: Suffragette(IIA) 106 mins Language: English Date: 16/11/2015(Mon) 13:20			
TicketId:3056 \$50(Student/Senior) House: B Seat: D3 FilmName: Suffragette(IIA) 106 mins Language: English Date: 16/11/2015(Mon) 13:20			

The final step is to implement the logout feature by creating a script file “**logout.php**”.

logout.php – it provides the following function (0.5 points):

- The page would destroy the current session, display the message “Logging out” (with <h2> tag) and redirect the user back to “index.html” (the login page) after 3 seconds.

Part 3 (Total: 12 points)

In this part, you are going to use CSS to decorate the pages which you have implemented in Part 2, as well as apply the responsive web design concept to **buywelcome.php**, **seatplantry.php**, **buyticket.php** and **confirm.php** pages. For the CSS implementation, it is regarded as the **design element** in the assignment which you are free to use any CSS formatting to make the web pages attractive and good-looking. While for the responsive web design (i.e. automatically resize, hide, shrink, or enlarge a website, to make it looks good on all devices (desktops, tablets, and phones)) in the abovementioned pages, you need to apply suitable settings to make the elements in the pages to be responsive to different mobile/computer screen resolutions in order to score the points. The table below illustrates the mark distribution in this part.

CSS:

Page	Point(s)
index.html, createaccount.html, main.php	1
comment.php	1
history.php	1

CSS+ responsive web design:

Page	Point(s)
buywelcome.php	3
seatplantry.php	4
buyticket.php	1

confirm.php	1
-------------	---

Submission:

The deadline of the Project I is on November 6, 2018 (Tuesday) 17:00.

You are required to:

1. Upload your work to your CS homepage at [https://i.cs.hku.hk/~\[your_CSID\]/project1/index.html](https://i.cs.hku.hk/~[your_CSID]/project1/index.html); this allows the tutors to check your project work easily.
2. Submit your work to the course Moodle submission page. To do that, you must copy all your HTML, CSS, JS, PHP and supporting files (e.g., images) by compressing or archiving all files to one package file and upload to the Moodle site. [Note: It would be nice if you can preserve the directory structure of your files in the archived or compressed file, e.g., you may place your images under the images/ folder and the CSS file under the style/ folder, etc. In the case, by simply decompress or retrieve the files, we can easily set up your Web site in another Web location. We shall provide an extra note to guide you to do this.]
3. Please indicate to us how much work you have completed for the project. You are recommended to make use of the table shown in the Grading Policy section as the checklist. Using it to show us how much you have accomplished. Please submit this checklist file to the course Moodle submission page too.

Grading Policy:

Part I (4 points)	<ul style="list-style-type: none"> • Correctness of "index.html" (2 points) • Correctness of "createaccount.html" (2 points)
Part II (12 points)	<ul style="list-style-type: none"> • Correct functionality of verifying user's login into the system (verifyLogin.php) (1 point) • Correct functionality of users' account creation (create.php) (1 point) • Correct functionality of the main page (main.php) (0.5 points) • Correct functionality of the welcome page in "Buy A Ticket" (buywelcome.php) (1 point) • Correct functionality of the seat selection page in "Buy A Ticket" (seatplantry.php) (2.5 points) • Correct functionality of buying a ticket page in "Buy A Ticket" (buyticket.php) (1 point) • Correct functionality of the confirm page in "Buy A Ticket" (confirm.php) (1 point) • Correct functionality of the comment page in "Movie Review" (comment.php) (2 points) • Correct functionality of retrieving the comment in "comment_retrieve.php" (0.5 points) • Correct functionality of submitting the comment in "comment_submit.php" (0.5 points) • Correct functionality of retrieving user's purchase history in "history.php" (0.5 points) • Correct functionality of logout in "logout.php" (0.5 points)
Part III (12 points)	<ul style="list-style-type: none"> • Use of CSS in the design of index.html, createaccount.html and main.php (1 point) • Use of CSS in the design of comment.php (1 point)

	<ul style="list-style-type: none"> • Use of CSS in the design of history.php (1 point) • Correct implementation of responsive web features and use of CSS in the design of buywelcome.php (3 points) • Correct implementation of responsive web features and use of CSS in the design of seatplantry.php (4 points) • Correct implementation of responsive web features and use of CSS in the design of buyticket.php (1 point) • Correct implementation of responsive web features and use of CSS in the design of confirm.php (1 point)
--	--

Plagiarism:

Plagiarism is a very serious academic offence. Students should understand what constitutes plagiarism, the consequences of committing an offence of plagiarism, and how to avoid it. **Please note that we may request you to explain to us how your program is functioning as well as we may also make use of software tools to detect software plagiarism.**