

**MODULAR PROGRAMME**

**COURSEWORK ASSESSMENT SPECIFICATION**

**Module Details**

|  |  |  |
| --- | --- | --- |
| **Module Code** UFCFR5-15-3 | **Run Main** | **Module Title**  Advanced Topics in Web Development 2 |
| **Module Leader** | | **Module Tutors SM LAU** |
| **Component and Element Number**  B (CW) | | **Weighting:** 50% |
| **Element Description** Coursework | | **Total Assignment time** |

**Dates**

|  |  |
| --- | --- |
| **Date Issued to Students** | **Date to be Returned to Students** |
| **Submission Place**  Soft copy : Moodlesp  (No need to submit hardcopy) | **Draft Submission Due:**  28-April-2024, 23:55 |
| **Final Submission Due:**  19-May-2024, 23:55 |

**Deliverables**

|  |
| --- |
| Working application and documentation as specified. |

**Module Leader Signature**

|  |
| --- |
|  |





**Assessed Coursework**

**Learning objectives**

* + Apply a range of web-oriented software architecture and design principles
  + Apply the principles of structured, functional & object-oriented programming
  + Apply good practices and testing in Web programming
  + Experience web application development with alternative paradigms and appreciate the pros and cons among them

**Overview**

In the module Advanced Topics in Web Development (ATWD1) in semester 1, you have developed a web application for search and maintenance of ATM information, using RESTful services and JavaScript-based AJAX clients. In this coursework, you are required to redevelop the client application using the MVC framework Angular.

You can make use of the RESTful server already developed in the module ATWD1; only the client-side application needs to be redeveloped. If you choose to redevelop the server, you are free to do so but no marks will be allocated to the server redevelopment.

You are also required to write a report to describe and compare the two different approaches for client-side application development, namely JavaScript-based AJAX and MVC framework, regarding but not limited to the followings: the development model and tools, strengths and weaknesses, and ease of maintenance.

You are strongly recommended to study the marks distribution and distribute your effort wisely.

NOTE: Extract of the coursework specification in ATWD is provided in the Appendix for your reference.

**Further technical requirements**

The following additional requirements must be strictly followed.

* + For the MVC approach, you **MUST** use Angular (http://angular.io) as the development framework. Your client application must include at least two Angular components in addition to the app component generated automatically.
  + Once again, development of the application using other frameworks such as React, VUE, etc. are strictly prohibited. By doing so, you will receive a ZERO mark for the entire coursework.

**User Interface and Experience**

* + You are free to use UI library at your own choice, such as semantic, bootstrap and Material UI. However, use of such libraries other than providing UI appearances are not allowed. As an example, form validations must be done using the built-in facilities provided by Angular Reactive Forms Module rather than by UI library. Refreshing and reloading of client pages must be severely limited. In other words, you are required to develop Single Page Applications (SPA) which is inherent to an Angular application.
  + You may design your own user interface style. However, the UI must be user-friendly and consistent in appearance in terms of font sizes, styles, colour usages, etc. The web pages must be neat and intuitive to use.
  + You have to design the operation flow very carefully. For example, a typical operation is to search for ATM machines in a specific district, then select one of the listed ATM for viewing the details. A very bad design is to request the user to remember the ID (or ATM name etc.) and go to another page for viewing the details by entering the ID (or ATM name etc.) again.

**Challenging Features**

* + While the aim of this coursework is to allow you experiencing alternative development approaches, you are encouraged to design and implement advanced features. You can document such features and illustrate them during the demonstration.

**Presentation, Demonstration and Documentation**

Presentation:

* + You are required to deliver a 5-minute PPT presentation on the mission, background, what you have done, tools and techniques employed, critical evaluation, further enhancement, etc.

Demonstration:

* + You are required to deliver a 10-minute demonstration on your application.
  + Note that it will be the students to lead the entire demonstration to show the functions and features rather than the marker to guide it.
  + Plan ahead your demonstration steps and highlight any special functional and design features.

Report:

* + You are required to submit a report to document your work and to evaluate the two development approaches. You are **free to structure the contents of your report**. However, the following sections MUST BE INCLUDED.
  + **Describe** the two approaches, JavaScript-based AJAX and MVC Framework: Description of the development approaches, such as its basic principles, development models, and development tools.
  + **Compare** the two approaches, JavaScript-based AJAX and MVC Framework: Critically compare the two approaches, including but not to limited to, pros and cons, strengths and weaknesses, and ease of maintenance.

**IMPORTANT NOTE ON REPORT WRITINGS**

After grading your ATWD1 reports, it has been found that the reports have lots of rooms for improvement in quality. You must read the reminders and suggestions below very carefully.

* + **Cover page** – A must, showing module code and module name, your name, SHAPE and UWE IDs, etc.
  + **Table of Contents** – You are recommended to add section and subsection numbering to help revealing the structure of your report. Each item in the Table of Contents must be associated with a page number.
  + **List of Figures** – If you report contains quite some figures, you may consider adding a List of Figures. Each figure needs to be numbered and has a caption. The caption of a figure appears at the bottom of the figure. In the List of Figures, show the number, caption and page number of each figure.
  + **List of Tables** – Similar to figures, but for tables. Note that the caption of a table appears at the top, rather than at the bottom.
  + **Background** – What is the intended purpose of the application? Overview on development approach and system architecture etc. Not to be very detailed about the development approaches as they will be covered in subsequent sections.
  + **Core contents** – Think carefully what are the important and essential contents to be included in the report. Arrange the contents with a logical structure. Separate the contents into multiple sections and sub-sections if needed. Read this coursework specification and the marking scheme very carefully and in detail.
  + **Conclusion** – A must. From Grammarly.com: “*Condensing all the points you’ve analyzed in a tidy little package*” and “*An effective conclusion paragraph should ultimately suggest to your reader that you’ve accomplished what you set out to prove.*” [Source: grammarly.com].
  + **References and in-text citations** – It won’t be possible for you to produce everything by yourself. You must have used other peoples’ software platform, idea, discussion, or whatsoever. Thus, you must include references and in-text citations to allow the readers to know the sources of those information you used and to allow them to dig out more if they wish. For referencing, you must employ the **UWE-Harvard Referencing Style.**
  + **No screen capture** – You will have the demonstration. You are not writing a user-guide. Thus, screen captures of your application are not appropriate core contents in your report. If you really wish, you may include them in the Appendix instead.

**Submission Guidelines**

Note that you should also submit the source code for the RESTful server developed in ATWD1 as a reference.

**Source Code:**

* Put all server code and supporting files into a folder name “server”.
* Put all client code (Angular project folder) and supporting files into a folder name “client”.
* Zip the above TWO folders into ONE single ZIP file. You must use the ordinary ZIP format; no rar or 7z allowed.
* Submit the ZIP file to Moodlesp before the deadline.
* NOTE: The submitted source codes **MUST BE THE SAME** as those used in the demonstration.

**Report:**

* Submit the report to Moodlesp before the deadline.
* Only MS-Word and PDF formats are allowed.

**Presentation PPT:**

* Submit the PPT file of your presentation to Moodlesp before the deadline.

**IMPORTANT:** It is the responsibility of students to make sure that the ZIP and other files can be extracted and opened successfully. Students will be awarded **ZERO** marks for the relevant component if the submitted file(s) cannot be opened successfully.

**Marks Distribution**

|  |  |
| --- | --- |
| Angular application development | 60% |
| Report | 30% |
| Presentation and Demonstration | 10% |

**NOTE**: Refer to the Marking Pro-forma on the next page for marking criteria. You are strongly recommended to read the marking pro-forma in details very carefully.

**Marking Pro-forma**

|  |  |  |
| --- | --- | --- |
| **Angular Application Development (Total 60%)** | | |
| Retrieval functions provided (30%) | 0-5 | Only a few trivial retrieval functions provided. |
| 6-10 | Support retrieval functions with different searching criteria. |
| 11-20 | Support retrieval functions and other update operations. |
| 21-25 | Support extensive retrieval functions and possibly other advanced features. |
| 26-30 | Exceptional piece of work with comprehensive functions, approaching a professional development. |
| Maintenance functions provided (10%) | 0-2 | Not all CUD (create, update, delete) operations are supported. |
| 3-5 | All CUD functions provided with fair quality. |
| 6-8 | All CUD functions provided with good quality, possibly a few functions for each operation. |
| 9-10 | Exceptional piece of work with sophisticated functions and very good UX design. |
| UX design (10%) | 0-2 | Raw, inconsistent and confusing UI. |
| 3-5 | Reasonable UI but without too much effort in workflow design. |
| 6-8 | Neat UI with consistent style. Users are able to finish a job intuitively.  Provides reasonable feedbacks to users when errors are returned from server. |
| 9-10 | Exceptional piece of work. Carefully designed to proactively assist a user to finish a job. Possibly with advanced UI features such as appropriate. |
| Implementation quality (10%) | 0-2 | Without error handling such as form validation. Possibly crash for some data input and work operations. |
| 3-5 | Some error handling and data validation but not sophisticated enough to ensure successful completion on an operation. |
| 6-8 | Extensive error handling and data validation. Robust operations. |
| 9-10 | Exceptional piece of work with very comprehensive error handling and data validation. Provide enough guidance and information to users should an error / invalid data input occurs. |

|  |  |  |
| --- | --- | --- |
| **Report (Total 30%)** | | |
| Descriptions on JavaScript-based AJAX approach (5%)   * Students are expected to discuss the philosophy and technical details behind and the pros and cons of the approach. | 0-1 | Only very brief overall on the approach. Without discussion on pros and cons. |
| 2-4 | Adequate description on the approach but not sophisticated nor in-depth enough. |
| 5 | Exceptional piece of work with critical analysis on the pros and cons of the approach. |
| Descriptions on MVC framework approach (10%)   * Students are expected to discuss the philosophy and technical details behind and the pros and cons of the approach. * Discussion on why Angular is an MVC framework. Architecture of an Angular application are expected. | 0-2 | Only very brief overall on MVC. Without discussion on pros and cons. |
| 3-5 | Adequate description on MVC and Angular provided but not sophisticated nor in-depth enough. |
| 6-8 | In-depth discussion on MVC. Detailed anatomy of an Angular application. |
| 9-10 | Exceptional piece of work with critical analysis on the pros and cons of MVC and Angular. |
| Comparisons between the two approaches (10%)   * Critically compare the two approaches. | 0-2 | Only very shallow comparison. Unable to identify what situation an approach can be applied effectively. |
| 3-5 | Some comparisons. Basic understanding on the applicability of each approach. |
| 6-8 | Good comparisons. With solid understanding on the situation/scenario when an approach can be applied effectively. |
| 9-10 | Exceptional piece of work. Able to demonstrate in-depth understanding on the two approaches. Critically evaluated and compared the two approaches. |
| Overall report quality (5%)   * The reports are expected to be formatted with a high standard and look professional. * Consistency in formatting like font faces, font sizes, paragraph spacing are expected. * All sections are expected to be numbered and reflected in the Table of Contents. * References are to be provided. | 0-1 | Poor report quality, missing some essential sections like table of contents and references. Without adequate effort in ensuring consistent and proper formatting. |
| 2-4 | Fair report quality with some efforts in ensuring consistent and proper formatting. |
| 5 | Exceptional quality with all essential sections completed at high quality. Comprehensive references. Almost free of grammatical mistakes. Proper use of **in-text citations.** |

|  |  |  |
| --- | --- | --- |
| **Presentation and Demonstration (Total 10%)** | | |
| Presentation (5%) | 0-2 | * Insufficient contents * Lack of preparation |
| 3-4 | * Adequate contents with essential items covered * Well prepared and smooth presentation |
| 5 | * Exceptional quality in both contents and presentation skills. Excellent selection of contents and PPT design. |
| Demonstration (5%) | 0-2 | * Poor organization and difficult to follow * Demo steps not planned in advance * Functions are not illustrated clearly |
| 3-4 | * Good organization with carefully planned steps to illustrate the functions |
| 5 | * Exceptional quality. Well planned and clear description to guide audience to understand the key functions intuitively and effortlessly. |

**Appendix**

**(Extract of coursework specifications in ATWD1 in Semester 1)**

**Overview**

You are required to build:

1. RESTful web services for search and maintenance of “Information of Automated Teller Machines of retail banks” provided by the Hong Kong Monetary Authority.
2. A web-based application as a client to the RESTful web services in (1). The client offers enquiries, visualization and maintenance of the information of ATM.

Additional requirements that must be fulfilled are:

* The data returned by the web services must be in JSON format. No client-side formatting instructions (such as HTML tags) are allowed to be included in the JSON returns. The RESTful web services must be implemented with PHP. For storing ATM information, you may use local MySQL database or a cloud-based database platform at your own choice.
* The client must employ significant JavaScript for AJAX and DOM manipulations so as to improve user experience. Refreshing and reloading of client pages must be severely limited. In other words, a Single Page Application (SPA) is expected.

This coursework consists of **FOUR** parts, which are specified in subsequent sections.

**Part 1: Data Conversion (10%)**

To set up your database of ATM information, you need to download the dataset from DATA.GOV.HK with the link:

<https://data.gov.hk/en-data/dataset/hk-hkma-bankbranch-banks-atm-locator>

The dataset is in JSON format. Meaning of the attributes are self-explanatory with the attribute names.

You are required to develop a server-side command tool to get the dataset from the link above and store the data into a database. Your tool must download, convert and store the data records on the fly without human intervention. In other words, you are **not** allowed to download and save the dataset as a local file, then convert and insert the records into the database by issuing separate commands. Instead of issuing commands, you may include the functionality in this part in your client application (see Part 3).

**NOTE:**

* The dataset is available in three languages: English, Traditional Chinese and Simplified Chinese. While it is mandatory for your web application to support English, you may opt to support Chinese too (see Section 3.4).
* Design the database schema carefully and make sure that it matches the dataset and supports the server APIs. You are advised to include a unique ID for every record.
* Shown below are the first few records and the last record in the dataset in English.

{

**"header":{**

**"success":true,**

**"err\_code":"0000",**

**"err\_msg":"No error found"**

**},**

**"result":{**

**"datasize":100,**

**"records":[**

{

"district":null,

"bank\_name":"Standard Chartered Bank (Hong Kong) Limited",

"type\_of\_machine":"Automatic Teller Machine",

"address":"Shops 106 - 109, 1/F, Mei Foo Plaza, Mei Foo Sun Chuen Stage 4, Kowloon",

"service\_hours":"24 Hours",

"latitude":"22.335983",

"longitude":"114.14140199999997"

},

{

"district":null,

"bank\_name":"Standard Chartered Bank (Hong Kong) Limited",

"type\_of\_machine":"Automatic Teller Machine",

"address":"Shop G6, G/F & Shop LG1, LG/F Westlands Gardens, 1027 King's Road, Quarry Bay, Hong Kong",

"service\_hours":"24 Hours",

"latitude":"22.28516",

"longitude":"114.21308099999999"

},

**……**

{

"district":"Yuen Long District",

"bank\_name":"Bank of China (Hong Kong) Limited",

"type\_of\_machine":"Automatic Teller Machine",

"address":"8-18 Castle Peak Road, Yuen Long, New Territories",

"service\_hours":"Mon - Fri: 9:00 – 17:00<br>Sat: 9:00 – 13:00<br>Sun and public holiday: Closed",

"latitude":"22.4443379",

"longitude":"114.0317002"

}

**]**

**}**

**}**

**Part 2: Design & Implementation of RESTful Web Services (Total 30%)**

You are required to develop RESTful web services that allow retrieval and maintenance of ATM records in your database constructed in Part 1. The data sent from your server to its clients must be in JSON format.

NOTE:

* PHP must be used in implementing the web services and no high-level framework such as CodeIgniter and Laravel can be used.

The web services must employ the HTTP basic methods as below:

|  |  |  |
| --- | --- | --- |
| HTTP methods | Usage | Marks |
| GET | **Retrieve** details of ATM records such as bank name, address, and service hours. | 10% |
| PUT | **Update** information of ATM records. | 5% |
| POST | **Add** new ATM records. | 5% |
| DELETE | **Remove** existing ATM records. | 5% |

**2.1 Further note on service invocations**

You must define the APIs (format of the URLs) precisely for invoking the web services.

* + For retrieval using GET, a possible format may be:

http://www.myserver.com/atm?district=Tsuen+Wan

The above URL requests the server to return the list of ATMs in the Tsuen Wan district.

* + It is your own discretion to decide whether parameter values are attached to the URL and sent through the HTTP body.

**NOTE:**

* + The API above is just an example. You may define your own API formats according to the operations and parameters needed.
  + The APIs must be precisely defined and documented in the report (see Part 4) or else developer of client applications will not be able to invoke the services.
  + To simplify your work, only English needs to be supported by your web services. However, your application may support Chinese as an advanced feature (see Section 3.4).

**2.2 Further note of data format**

Any data returned from the server to clients must be in JSON format. Shown below is a possible return from the server when the client has requested a list of ATMs of the Bank of China in the Yuen Long district.

{

"header":{

"success":true,

"err\_code":"0000",

"err\_msg":"No error found"

},

"result":[

{

"district":"Yuen Long District",

"bank\_name":"Bank of China (Hong Kong) Limited",

"type\_of\_machine":"Automatic Teller Machine",

"address":"8-18 Castle Peak Road, Yuen Long, New Territories",

"service\_hours":"Mon - Fri: 9:00 – 17:00<br>Sat: 9:00 – 13:00, Sun and public holiday: Closed",

"latitude":"22.4443379",

"longitude":"114.0317002"

},

{

"district":"YuenLong",

"bank\_name":"OCBC Bank (Hong Kong) Limited",

"type\_of\_machine":"Automatic Teller Machine",

"address":"Shop 1-3, G/F, 40-54 Castle Peak Road, Yuen Long",

"service\_hours":"7x24",

"latitude":"22.444222",

"longitude":"114.030768"

},

{

"district":"YuenLong",

"bank\_name":"Industrial and Commercial Bank of China Limited",

"type\_of\_machine":"Automatic Teller Machine",

"address":"Shop No. 311, Level 3, Ticketing Concourse, KCR Lok Ma Chau Station, N.T (Paid Concourse)",

"service\_hours":"Follows the service hours of the mall or MTR station where it locates",

"latitude":"22.514434",

"longitude":"114.065658"

},

……

]

}

**NOTE:**

* + The above JSON return is just an example. You may define your own JSON format.

**2.3 Error Handling (5%)**

The web services are required to handle errors in a graceful manner. When an API is invoked incorrectly (for example, missing an essential parameter or having incorrect parameter format), the server must return relevant and meaningful error codes and messages to the client. Again, the error code and message are formatted as JSON strings. An example is shown below where a parameter is invalid for a retrieval service.

http://www.myserver.com/atm/district=

|  |
| --- |
| {  "header":{  "success":false,  "err\_code":"0120",  "err\_msg":"missing district value "  }  } |

You need to define your own error codes and messages according to your own web services. The error codes and messages must be precisely defined and documented in the report (see Part 4).

**NOTE:**

* + The above JSON return is just an example. You may define your own JSON format.

**Part 3: Client-side Application (Total 30%)**

**3.1 Web application to enable retrieval of information on ATM records (10%)**

Design and implement a Single Page Application (SPA) that uses the RESTful services implemented in Part 2. Your application should provide an interface for users to enter the retrieval criteria like service hours and location district.

As a SPA, refreshing and reloading of the entire page is prohibited. Instead, JavaScript, AJAX and DOM manipulations should be used to make requests, retrieve server replies, and updates web contents for displaying retrieval results.

**NOTE:**

* + The client-side application must be written in HTML and JavaScript.
  + Use of UI libraries such as Semantic UI and Bootstrap are allowed but only limited for providing UI effects and input form validations.
  + The use of jQuery is only limited for UI effects. The use of jQuery for making AJAX requests to server APIs is strictly prohibited.
  + The use of high-level frameworks for application generation or completing a server request cycle is strictly prohibited.

**3.2 Web application to maintain information on ATM records (10%)**

Your application should provide web pages to allow maintenance of ATM records including addition of a new ATM, removal of an existing ATM, and update of ATM details. The maintenance functions must employ the RESTful services implemented in Part 2.

**NOTE**:

* + The pages for maintenance of ATM records can be separated from the retrieval page in (3.1) as they are essentially different functionalities. However, excessive refreshing and reload should still be avoided for ensuring better user experience.

**3.3 Usability and User Experience (5%)**

You have to design the operation flow very carefully. For example, a typical operation is to search for ATMs in a specific district, then select one of the listed ATMs for viewing the details. A very bad design is to request the user to remember the ID or other fields and go to another page for viewing the details by entering the ID or other fields again.

**3.4 Challenging features (5%)**

You are encouraged to design and implement challenging features besides those stated above, such as locations of ATMs on a Google Map, path going to a selected ATM, support of Chinese, etc.

**NOTE**:

* + When you design the user interface in the client application, make sure that it is user-friendly and is consistent in its appearance in terms of styles, font sizes, and the use of colours. The operations must also be intuitive to novice users.

**Part 4: Presentation, Demonstration and Documentation (30%)**

**Presentation (5%):**

* + You are required to deliver a 5-minute PPT presentation on what you have done, the design of the server API, tools and techniques employed, critical evaluation, further enhancement, etc.

**Demonstration (5%):**

* + You are required to deliver a 10-minute demonstration on your application, including the server-side data conversion program, the server APIs and the client-side application.
  + For the server APIs, you need to use **Postman or similar tools** to make requests to the server and to show the JSON returns from the server, including both normal data and error codes/messages.
  + Note that it will be the students to lead the entire demonstration to show the functions and features rather than the marker to guide it.

**Report (20%)**

* + You are required to submit a written report to document the design and technical details of your application.
  + While you are the one to decide the structure and contents of the report, the following items MUST BE included.
  + Design & Implementation, including specification of APIs to invoke your web services and the error codes and messages possibly returned by the server.
  + Explain clearly how different software technologies (e.g. XML, XSLT, CSS, JavaScript, Object-Oriented programming and design patterns, etc.) were applied in your system.
  + Evaluation of your application showing the advantages over the traditional non-RESTful systems.
  + Future improvement
  + Conclusion

**END.**