

**MODULAR PROGRAMME**

**COURSEWORK ASSESSMENT SPECIFICATION**

**(DRAFT)**

**Module Details**

|  |  |  |
| --- | --- | --- |
| **Module Code** UFCE3Q-30-3 | **Run Main (AY25/26 Sem 1)**  **Full-Time Cohort** | **Module Title**  Advanced Web Development |
| **Module Leader** | | **Module Tutors SM LAU** |
| **Component and Element Number**  Individual Portfolio | | **Weighting:** 100% |
| **Element Description** Build a web application demonstrating the ability to apply advanced web development techniques in a real-world context. | | **Total Assignment time** |

**Dates**

|  |  |
| --- | --- |
| **Date Issued to Students**  7-Sep-2025 | **Date to be Returned to Students** |
| **Submission Place**  Soft copy : Moodle | **Draft Submission Date and Time:**  2-Nov-25, Sunday, 23:55 |
| **Submission Date and Time:**  7-Dec-25, Sunday, 23:55 |

**Deliverables**

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

**Module Leader Signature**

|  |
| --- |
|  |

**Overview**

For this **individual** assignment, students will design, build and deploy a web application demonstrating their ability to apply advanced web development techniques in a real-world context. Documentation to critically evaluate the application’s design, implementation, performance and adherence to web standards and best practices will also be produced as part of the portfolio. Through this individual assignment, the following Module Learning Outcomes are assessed:

* **MO1** Analyse and evaluate web standards, communication protocols and emerging technologies, demonstrating the ability to apply object-oriented programming techniques in web application development.
* **MO2** Recognise and apply common software patterns, and web architectures in practice.
* **MO3** Demonstrate proficiency in using contemporary tools, techniques, and web frameworks throughout the web development project lifecycle.
* **MO4** Employ effective development methods, testing strategies and software documentation practices to create and critique web applications, showcasing an understanding of the importance of these practices in the development process.

**Tasks to do**

You are required to:

1. build a server providing RESTful web services for search and maintenance of “Mobile Post Office” provided by the data.gov.hk,
2. build a web-based application as a client to the RESTful web services in (1). The client offers enquiries, visualization and maintenance of the information,
3. perform, document, and analyse a series of testing for an objective evaluation on the applicability, usability and robustness of the entire application stack, including the data storage, web services, and the client application,
4. produce a detailed documentation regarding the application architecture, tools and platforms employed, web services and client-side functions offered, evaluation on the design and development, suggestions on further enhancement, as well as your own reflection on the entire project cycle, and
5. deliver a presentation and demonstration to showcase your work and application product.

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

This is an **individual** assignment.

**Part 1: Data Conversion (3%)**

To set up your database of Mobile Post Office information, you need to download the dataset in JSON format with the link:

<https://data.gov.hk/en-data/dataset/hk-hkpo-hkpo_ds01-hkpo-mobile-office>

You are advised to have a look at the data dictionary provided with the link above to understand the data fields and JSON structure. Shown below are the first two records in the dataset.

**{**

**"data":[**

**{**

**"mobileCode":"1",**

**"locationTC":"深井",**

**"locationSC":"深井",**

**"addressTC":"青山公路深井段，海韻花園外。",**

**"nameSC":"流动邮政局 1",**

**"districtSC":"荃湾区",**

**"addressSC":"青山公路深井段，海韵花园外。",**

**"closeHour":"09:30",**

**"nameTC":"流動郵政局 1",**

**"districtTC":"荃灣區",**

**"latitude":"22.36774",**

**"openHour":"09:00",**

**"dayOfWeekCode":1,**

**"nameEN":"Mobile Post Offices 1",**

**"districtEN":"Tsuen Wan",**

**"locationEN":"Sham Tseng",**

**"addressEN":"Close to Rhine Garden, Castle Peak Road (Sham Tseng)",**

**"seq":1,**

**"longitude":"114.06233"**

**},**

**{**

**"mobileCode":"1",**

**"locationTC":"黃金海岸",**

**"locationSC":"黄金海岸",**

**"addressTC":"屯門黃金海岸，黃金海岸商場外。",**

**"nameSC":"流动邮政局 1",**

**"districtSC":"屯门区",**

**"addressSC":"屯门黄金海岸，黄金海岸商场外。",**

**"closeHour":"10:05",**

**"nameTC":"流動郵政局 1",**

**"districtTC":"屯門區",**

**"latitude":"22.37184",**

**"openHour":"09:50",**

**"dayOfWeekCode":1,**

**"nameEN":"Mobile Post Offices 1",**

**"districtEN":"Tuen Mun",**

**"locationEN":"Hong Kong Gold Coast",**

**"addressEN":"Outside Gold Coast Piazza, Gold Coast, Tuen Mun.",**

**"seq":2,**

**"longitude":"113.99305"**

**},**

**…**

**]**

**}**

You are required to develop a server-side program to get the dataset from the link above and migrate the data into a database. You may first download the dataset, then migrate the data to the database. Alternatively, you may download and migrate the data to the database on-the-fly using one single step.

**NOTE:**

* The dataset contains both Traditional Chinese, Simplified Chinese and English contents. While it is mandatory for your web application to support English, you may opt to support Traditional Chinese and Simplified Chinese too (see Section 3.3).
* Design the database schema carefully and make sure that it matches the dataset and provide enough support to the server APIs. **You are advised to include a unique ID for every Mobile Post Office record and to use the ID to identify the target record for CRUD operations.**

**Part 2: Design & Implementation of RESTful Web Services (Total 22%)**

You are required to develop RESTful web services that allow retrieval and maintenance of Mobile Post Office records in your database constructed in Part 1. The data sent from your server to its clients must be in JSON format. No client-side formatting instructions (such as HTML tags and CSS rules) are allowed to be included in the JSON returns.

PHP or NodeJS must be used in implementing the web services and no high-level framework such as CodeIgniter and Laravel can be used. For storing Mobile Post Office records, you may use local MySQL database or a cloud-based database platform at your own choice.

The web services must employ the HTTP methods as below:

|  |  |  |
| --- | --- | --- |
| HTTP methods | Usage | Marks |
| POST | **Create** new records. | 3% |
| GET | **Retrieve** details of records such as district, address and opening hours. | 9% |
| PUT | **Update** information of records. | 4% |
| DELETE | **Delete** existing records. | 3% |

**2.1 Number of APIs to be supported**

While you are highly encouraged to produce comprehensive APIs to support different client requests, you may also refer to the table below that shows the minimum number and requirements of the APIs that MUST be provided by the server for different kinds of operations.

|  |  |  |
| --- | --- | --- |
| API Type | Minimum number of API(s) | Remark |
| Create | 1 | * Create a Mobile Post Office record. * The data values must sent through HTTP request body. |
| Retrieve | 3 | * Allow the client to search a record using different criteria, for example,   + retrieval by using ID of a record,   + search the Mobile Post Offices in a particular district,   + etc. * There MUST be an API that supports retrieval of records using multiple search criteria, for example, search of Mobile Post Offices in the Tsuen Wan District that opens at 14:00. |
| Update | 1 | * The API MUST support partial updates of a record – only data values of those fields that need to be updated are sent to the server. In other words, the server should not “overwrite” an entire record but just to update individual fields that are to be changed. |
| Delete | 1 | * Delete a Mobile Post Office record. * The identifier for specifying the record to be deleted can be sent through URL or HTTP request body. |

**2.2 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/mobilepost?district=Yuen+Long

The above URL requests the server to return the list of Mobile Post Office in the Yuen Long district.

* + It is your own discretion to decide whether parameter values are attached to the URL and sent through the HTTP body in GET and DELETE operations.
  + For PUT and POST operations, parameter values MUST be 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 5) or else developer of client applications will not know how to invoke the services.
  + To simplify your work, only English needs to be supported by your web services. However, your application may support Traditional Chinese and Simplified Chinese as an advanced feature (see Section 3.3).

**2.3 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 Mobile Post Office in the Yuen Long district. The JSON return is just an example. You may define your own JSON format.

{

"header":{

"success":true,

"message":"12 records retrieved",

"err\_code":"0000",

"err\_msg":"No error found"

},

"result":[

{

"mobileCode":"1",

"locationTC":"洪水橋",

"locationSC":"洪水桥",

"addressTC":"元朗洪水橋大路與青山公路洪水橋段交界，翠珊園附近。",

"nameSC":"流动邮政局 1",

"districtSC":"元朗区",

"addressSC":"元朗洪水桥大路与青山公路洪水桥段交界，翠珊园附近。",

"closeHour":"14:40",

"nameTC":"流動郵政局 1",

"districtTC":"元朗區",

"latitude":"22.43353",

"openHour":"14:10",

"dayOfWeekCode":1,

"nameEN":"Mobile Post Offices 1",

"districtEN":"Yuen Long",

"locationEN":"Hung Shui Kiu",

"addressEN":"Close to Chui Shan Yuen, at the junction of Hung Shui Kiu Main Road and Castle Peak Road (Hung Shui Kiu), Yuen Long.",

"seq":7,

"longitude":"113.99781"

},

{

"mobileCode":"1",

"locationTC":"洪水橋",

"locationSC":"洪水桥",

"addressTC":"元朗洪水橋大路與青山公路洪水橋段交界，翠珊園附近。",

"nameSC":"流动邮政局 1",

"districtSC":"元朗区",

"addressSC":"元朗洪水桥大路与青山公路洪水桥段交界，翠珊园附近。",

"closeHour":"15:25",

"nameTC":"流動郵政局 1",

"districtTC":"元朗區",

"latitude":"22.43353",

"openHour":"14:55",

"dayOfWeekCode":3,

"nameEN":"Mobile Post Offices 1",

"districtEN":"Yuen Long",

"locationEN":"Hung Shui Kiu",

"addressEN":"Close to Chui Shan Yuen, at the junction of Hung Shui Kiu Main Road and Castle Peak Road (Hung Shui Kiu), Yuen Long.",

"seq":7,

"longitude":"113.99781"

},

……

]

}

**2.4 Error Handling (3%)**

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 in JSON. An example is shown below when a parameter is invalid for a retrieval service.

http://www.myserver.com/mobilepost/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 5). The above JSON return is just an example. You may define your own JSON format

**Part 3: Design & Implementation of Client-side Application (Total 30%)**

**3.1 Web application to enable retrieval and maintenance of Mobile Post Office records**

Design and implement a client-side web application for retrieval and maintenance of Mobile Post Office records using the APIs provided by your server in Part 1&2. You must use the **Angular Framework** as the development platform of the client application. Development of the application using other frameworks such as React, NextJS, VUE, etc. are strictly prohibited. By doing so, you will receive a ZERO mark for the entire coursework. The Angular client application must include at least FOUR Angular components in addition to the app component generated automatically.

**NOTE:**

* + Use of UI libraries such as Semantic UI and Bootstrap are allowed but only limited for providing UI effects.
  + The use of Materials UI is recommended but not mandatory.
  + The use of jQuery is only limited for UI effects. The use of jQuery to perform form validation and to make AJAX requests to server APIs are strictly prohibited.

**3.2 User Interface and Experience**

* + You are free to use UI library at your own choice, such as semantic, bootstrap and Material. 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 with proper design and project structure.
  + 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 Mobile Post Office in a specific district, then select one of the listed records for viewing the details. A very bad design is to request the user to remember the ID and go to another page for viewing the details by entering the ID again.
  + The pages for maintenance of Mobile Post Office records can be separated from the retrieval page(s) as they are essentially different functionalities. However, excessive refreshing and reload should still be avoided for ensuring better user experience.

**3.3 Challenging Features**

* + You are encouraged to design and implement advanced features such as Google Map and Path, multi-lingual support, responsive pages, etc. You can document such features and illustrate them during the demonstration.

**Part 4: Testing (10%)**

You need to design and perform different tests to verify that your web application fulfils the expected functionalities with appropriate level of quality. Both the RESTful web services and the client application are to be tested. You need to produce a **separate test report** to detail the tests performed, including

* + test strategy, test plan, test cases, expected outcome,
  + actual results and their analysis, interpretations on the results, and the insights that arose.

Your test plan should at least covers two aspects: (1) correctness and robustness, and (2) user acceptance (i.e. you need to perform a User Acceptance Testing, UAT). For the UAT, the portfolio of the human subjects should be included with justification why the subjects are valid and representatives as being the target users.

You may consider putting the raw results as an appendix to avoid occupying too much space in the main report.

**Part 5: Presentation, Demonstration, Documentation and Reflection (35%)**

**Presentation:**

* + You are required to deliver a 5-minute PPT presentation on the project background and objective, and what you have done, such as architecture, design of the server APIs, design of the client application, tools and techniques employed, critical evaluation, further enhancement, etc.
  + Marks will be awarded according to what you have done in building the web application, and most importantly your critical reflection and evaluation for the entire development process.
  + DO NOT describe the functionalities of your client application with lots of screen captures. This is not necessary as you have the demonstration session. Instead, you should focus on high level issues such as architecture, design philosophy, evaluation and reflection etc.
  + Although you will not be assessed with your presentation skills and visual quality of your presentation slides, as a degree level student, you are expected to have a high quality and well-prepared presentation. In particular, reading scripts, no matter on paper or with any digital device, is strictly prohibited.

**Demonstration:**

* + You are required to deliver a 25-minute demonstration on your application, including (1) the server-side data conversion program, (2) the server APIs, and (3) the client-side application.
  + For the data conversion, you need to start with an empty database and to show how your program migrates the data to the database. Database contents before and after the conversion are to be demonstrated.
  + For the server APIs, you need to use **Postman or similar tools** to make requests to the server for CRUD operations and to show directly 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.
  + Plan ahead your demonstration steps and highlight any special functional and design features.
  + Marks will be awarded according to the comprehensiveness of the APIs and the associated error handling, the functionality and usability of the client application, and also your reflection to the entire development process as exhibited through questions and answers.

**Q&A:**

* + After the presentation and demonstration, there will be a 10-minute Q&A session, during which the markers will raise questions to clarify your work and solutions, and to provide opportunities for you to further elaborate your own thought and reflection through working on the project.
  + Note that the Q&A session is very important for you to demonstrate your own critical thought, thus will contribute a significant amount in the overall mark in this assessment.

The exact date and time for the presentation, demonstration and Q&A will be announced in due course. Tentatively, it will be held on the week of 7-Dec-2025.

Make sure that you join your session on time and have everything ready. The marker will not wait for you to set up and load the presentation and demonstration materials. You must observe the time used very carefully. No extra time will be given if you fail to cover all your contents within the specified time limit. Likewise, if you join the session late, you waste your own time and no extra time will be given.

**Project Report:**

* + You are required to submit a written report to document the design and technical details of the server APIs and the client application. Think carefully what important and essential contents are 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.
  + While you are the one to decide the structure and contents of the report, the following items MUST BE included. (Note carefully that the list shows what you need to include in your report, not necessarily the report structure. You may design you own report structure to suit your own writing needs.)
  + **Cover page** – A must, showing module code and module name, names and IDs of group members, 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 problem scenario? What is the intended purpose of the application? What have you done in this project?
  + **Software Patterns and Web Application Architecture Employed**
    - Can you name and describe different web application architectures?
    - What architecture do you have employed in your work, and why?
    - Can you name and describe the software patterns that you have employed in your work?
    - How the architectural components interact to achieve the ultimate goal of the web application?
  + **Development Tools, Libraries and Frameworks Employed**
    - What development tools, libraries and frameworks do you have employed?
    - What are the features of such tools, libraries and frameworks?
    - Why do you have selected them for your project?
    - What are the benefits of using such tools, libraries and frameworks over developing a web application with low level programming constructs such as plain JavaScript combined with HTML/CSS as separate and isolated programming units?
  + **RESTful APIs Specification**, including both the request and response formats and specification of error codes. If you have lots of APIs to define, you may consider providing the full API specifications as an Appendix, and to include a brief summary in the core report content.
  + **Specification of client-side functions** provided by the client application.
  + **Evaluation of your product** – Think critically the strengths and weaknesses of your web application from different aspects, including but not limited to the comprehensiveness of the server APIs, the functionalities and usability of the client applications, pros and cons of your web architecture, strengths and weaknesses of the development tools and platforms that you have chosen, the experience that you have learned about web application development through working on this project, etc.   
      
    **REMINDER**: Evaluation means that you need to have your own critical thinking and reflection in the context of the work that you have produced in this project, rather than just giving factual points (such as description on what Angular is, what RESTful approach means, etc.). You need to share your experience and lessons learned through working on this project. Providing only factual contents will result in a very low mark or even not been awarded with any marks at all.
  + **Suggestions on future improvement**, including both the server and the client application, and possibly any suggestions on how the overall application architecture can be improved. Rationale should be given for each suggestion explaining why such suggestions are worthwhile to be produced in the future.   
      
    **REMINDER**: The suggestions on future improvement should be in the context of the problem scenario and be relevant to your work, rather than just some very generic and vague descriptions. Bad examples are (1) “to enhance the user-friendliness” but without telling how, (2) “to enhance the security” without telling how and why, etc.
  + **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.** Link below.

[**https://www.uwe.ac.uk/study/study-support/study-skills/referencing/uwe-bristol-harvard**](https://www.uwe.ac.uk/study/study-support/study-skills/referencing/uwe-bristol-harvard)

* + Further Note on Report:
  + **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.
  + **No source listing** – There is no need to dump the source code, neither as the core contents nor in the appendix. If you wish to use some program codes to help illustrating the contents, you may do so, but strictly limit the amount and make sure that the codes are for illustrating some high level concepts only.

**Draft Submission**

In order to ensure that you have proper progress and your work is on the right track, you are required to submit a draft of your report and a video record of your prototype demonstration by **2-Nov-2025, 23:55** to Moodle.

Note that the draft submission is mandatory and will be proven to be very useful for you to ensure that your work is in the correct direction and to polish your work before the final submission. Experience in previous cohorts is that students who had not submitted the draft usually turned out to produce reports with inappropriate contents, thus received a very low mark. Take your own risks if you do not submit the draft.

**Submission Guidelines**

You have to submit FIVE items listed below.

(1) Presentation PPT

(2) Server Web Services - Source Code and supporting files

* Submit all source files including SQL for creating and manipulating the database, and all necessary files to build and deploy the web services. If you have employed any external libraries, they must also be submitted.

(3) Client-side Application – Source Code and supporting files

* Submit all source files including the Angular project and all necessary files to build and deploy the client application. If you have employed any external libraries, they must also be submitted.

For (2) and (3), compress all files and folders into one single ZIP file (zip only, rar and 7zip are NOT allowed). The submitted source codes **MUST BE THE SAME** as those used in the demonstration.

(4) Test Report

* Only MS-Word or PDF formats are allowed.

(5) Project Report

* Only MS-Word or PDF formats are allowed.

For (4) and (5), if you use any online editing tools, make sure that the exported files contain proper formatting and line spacing.

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

**Assessment Criteria**

|  |  |
| --- | --- |
| **Part 1: Data Conversion (3%)** | |
| 0 | * Failed to download data source and migrate to MySQL database or any cloud-based database platforms. |
| 1 | * Able to download data source and migrate to MySQL database or any cloud-based database platforms. * Some deficiencies in the process, for example, not on the fly but with intervention with separate command steps. * Database schema may not be able to incorporate all fields in the data source. |
| 2 | * Successful migration of data to MySQL database or any cloud-based database platforms on the fly using one single command. * Good database design that supports all fields in the data source. |
| 3 | * Excellent implementation quality, possibly with some advanced features, for example, report on data items imported or any irregularities. |

|  |  |  |
| --- | --- | --- |
| **Part 2: Design & implementation of RESTful Web Services (Total 22%)** | | |
| Retrieval  (9%) | 0-2 | * Partially functional with errors when retrieving information on records. |
| 3-4 | * Functional but only very limited or no searching criterion is supported. |
| 5-7 | * Support different search criteria * Reasonable API format * Support some error checking * Reply to client application in JSON format |
| 8-9 | * Sophisticated search criteria * Good API format |
| Update  (4%) | 0-1 | * Partially functional with errors in updating records. * NOT PUT – Award 0 mark if not using the PUT method, regardless of other assessment criteria in this component |
| 2-3 | * Using PUT method * Reasonable API format * Support some error checking |
| 4 | * Good API format * Only fields to be updated are sent to server |
| Add  (3%) | 0-1 | * Partially functional with errors in adding new records. * NOT POST – Award 0 mark if not using the POST method, regardless of other assessment criteria in this component. |
| 2-3 | * Using POST method * Reasonable API format * Support some error checking |
| Remove  (3%) | 0-1 | * Partially functional with errors in deleting records. * NOT DELETE – Award 0 mark if not using the DELETE method, regardless of other assessment criteria in this component. |
| 2-3 | * Using DELETE method * Reasonable API format * Support some error checking |
| Error Handling  (3%) | 0-1 | * No or only very limited error codes and messages |
| 2-3 | * Adequate to sophisticated error codes and messages * Well design of error codes (e.g. hierarchical numbering according to error types) * Clear and meaningful error messages that describe the nature and cause of errors clearly |

|  |  |  |
| --- | --- | --- |
| **Part 3: Client-side Angular application development (Total 30%)** | | |
| Retrieval of information on records  (10%) | 0-2 | * Partially functional with errors when retrieving records. |
| 3-5 | * Functional and have reasonable UI for specifying searching criteria * NOT SPA – Award max 5 marks if not being SPA, regardless of other assessment criteria in this component. |
| 6-8 | * SPA * Good UI that allows users to specify searching criteria easily and precisely * Provides reasonable feedbacks to users where errors are returned from server. |
| 9-10 | * SPA * Sophisticated search criteria * Clear and intuitive formatting of record details possibly with advanced features like using JQuery for enhancing visual effects. |
| Maintenance of records  (8%) | 0-2 | * Partially functional with errors |
| 3-6 | * Functional and have reasonable UI * Some error checking and feedbacks to users when errors occur |
| 7-8 | * Provide adequate responses to users after successful completion of maintenance operations * Sophisticated error checking and feedbacks to users when error occur |
| Usability & User Experience (5%) | 0-1 | * Poor workflow design * Not intuitive to users for performing the operations. |
| 2-3 | * Good UI and workflow design * Straight forward and intuitive for users to perform required operations. |
| 4-5 | * Extensive effort in UI and workflow design, leading to very good usability and user experience. * Consistent and stylish design, probably with some advanced features. |
| Extra and challenging features  (7%) | * Award marks according to usefulness and implementation difficulty * Each feature should normally be awarded with 2 marks and max 3 marks * Examples: multi-lingual support, map and path, pagination and sorting of records, responsive pages | |

|  |  |  |
| --- | --- | --- |
| **Part 4: Testing (Total 10%)** | | |
| Testing steps and execution  (5%) | 0-1 | * Only a few trivial test cases performed, without interpretation or even raw results. |
| 2-3 | * A number of test cases performed but not comprehensive enough. |
| 4-5 | * Rather comprehensive testing performed covering different aspects. |
| Test documentation and analysis  (5%) | 0-1 | * Little and weak analysis and interpretation on the results. |
| 2-3 | * Raw results are documented in detail with some analysis and interpretation of results. |
| 4-5 | * Comprehensive analysis and interpretation leading to insights to the product quality, weakness and enhancement needed. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Part 5: Presentation, Demonstration, Documentation and Reflection (Total 35%)**  \*The presentation and demonstration serve to demonstrate students’ proficiency in connection to the Module Learning Outcomes. Markings are based on the contents rather than the presentation and demonstration skills. | | | |
| Software Patterns and Architecture (5%) | 0-1 | | * Only very limited or no information regarding different types of web architectures. * Not able to show the web architecture of student’s own work. |
| 2-3 | | * Able to identify and compare different web architectures. * The web architecture of the web application documented clearly with clear indications on the components and their interactions. |
| 4-5 | | * In-depth discussion on the pros and cons for the architecture adopted, possibly also stating current and future trends in advanced web application architecture. |
| Development tools, libraries, and frameworks  (5%) | 0-1 | | * Only stating some trivial tools (such as IDE) without describing their roles in the project. |
| 2-3 | | * Able to identify high level tools, libraries and frameworks employed in the project with some brief discussion. |
| 4-5 | | * Comprehensive discussion on the tools, libraries and frameworks, particularly how they can facilitate the development process, enable realization of the web architecture, and ensure future maintenance. |
| Web service API specifications  (5%) | 0-1 | | * Insufficient details provided for API formats. * Poor API design that may not support necessary parameters |
| 2-3 | | * Good API design with clear and precise specification on formats * Sufficient details provided for API users |
| 4-5 | | * Comprehensive API spec with all necessary details. * Having good readability for readers to grasp the APIs very easily. |
| Specification of functions  (5%) | 0-1 | | * Only a flat list of functions without much description or justifications |
| 2-3 | | * Categorized list of functions with sufficient descriptions and justifications * The list of functions may not be comprehensive enough and only rather coarse functions provided. |
| 4-5 | | * Comprehensive list of functions at appropriate granularity, each with appropriate descriptions and justifications. |
| Evaluation  (5%) | 0-1 | | * Only factual points without addressing the development work or the product. |
| 2 | | * A few relevant points based on actual experience and critical evaluation but not quite comprehensive. |
| 3-4 | | * A good discussion based on the experience and lessons learned through this project. * Contents are addressing the experience in this project rather than just being generic. |
| 5 | | * Excellent discussion demonstrated that the student have in-depth critical thinking on the entire development cycle and able to have reflection after working on the project. * Contents are addressing the project rather than just being generic. |
| Discussion of future improvement  (5%) | 0-2 | | * Only a few trivial points without much description. |
| 3-4 | | * Quite some meaning improvements suggested with sufficient description to justify why the improvements are worthwhile to consider. |
| 5 | | * Comprehensive suggestions covering different aspects and providing detailed descriptions and justifications. |
| Formatting and overall report quality  (5%) | 0-2 | * Incomplete report or only very little details provided. * Missing or a few references and in-text citations. | |
| 3-4 | * Appropriate level of details provided but fair organization and formatting * Some references and in-text citations provided but not comprehensive enough | |
| 5 | * Very good organization and formatting with sufficient details * Extensive references and in-text citations. | |

**END. (Draft)**