**Diploma in Software and Design**

**Assignment Cover Sheet**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Student’s name:** AMARBIR SINGH | | | | |
| **Module 6 Major Project / Internship** | | | | |
| **Assignment title and/or number**:  DSED-08 Major Project | | | | |
| **Assessment weighting** | | | *Need to pass the assessment to complete the course* | |
| **Passing Criteria:** | | | Need to score 50 or more marks to pass the assessment.  **Total Marks : 100** | |
| **Due date**: Wednesday March 25, 2020 | | | **Date submitted**:  (late submissions incur 10% penalty, after 7 days late, the assessment will not be marked) | |
| **Assessment conditions:** | | | This is a resource-based assessment. This means that you may have access to any relevant resources to assist you. This could include, for example, your learning materials, information on the Internet, and so on. However, all work must be your own with no assistance from any other person. | |
| **Submission requirements:** | | | You’re required to upload the following on Cloud Campus:   * This document, completed where appropriate * Visual Studio project files * Upload your project on Github and paste the link below   GitHub Link: <https://github.com/amar751/OnlineEducationPortal> | |
| **Learning Outcomes:** | * Applying appropriate business process modelling tools to analyse and document business processes; * Software estimation and metrics; * Creating accurate and clear technical and user documentation; * Application of the core interaction design concepts and practice, underpinned in the third outcome of the New Zealand Certificate in Information Technology (Level 5)[Ref: 2595]; * Application of the core information systems skills and knowledge underpinned in the second outcome in the New Zealand Certificate in Information Technology [Ref: 2595]. * Coding – object oriented, procedural; * Construct software with complex, multi-element architectures and abstract data types (ADTs), such as general graphs, trees, tables; * Writing code following design patterns and software development standards * Source and version control; * Optimisation concepts and techniques; * Application of the core software development concepts and practice, underpinned in the fourth outcome of the New Zealand Certificate in Information Technology (Level 5) [Ref: 2595]. * Designing a variety of tests including unit and system tests, usability testing, user acceptance tests; incorporating a range of testing techniques e.g. white box, black box, boundary-value testing; * Designing and implementing appropriate application data access, management, and storage technologies to match the application domain; * Digital asset management and storage technologies appropriate to match the application domain e.g. source and version control, artefact repository * Application security principles, including current best practices in IT security e.g. OWASP; * Encryption and privacy; * Tool selection and architecture; * Understanding service orientation and using external services e.g. simple object access protocol (SOAP), representational state transfer (REST); * Software architectural patterns including model view controller (MVC) and model view presenter (MVP). * Critical thinking, business logic, organisational processes, innovation and enterprise skills; * Project planning, management and control – cost, risk, quality, stakeholder, change, configuration, contracts, and maintenance management * Application of professional and ethical practice, including sustainability, equity, social and contemporary cultural issues, relevant to an IT organisational environment (e.g. Treaty of Waitangi and accessibility issues); * Organisational implications of managing and complying with legal and regulatory requirements (e.g. health and safety, contract management, licensing, privacy; observing security responsibilities and industry codes of practices, and codes of conduct (e.g. IITP) relevant to an organisational environment. * Information representation design for multiple situations e.g. data visualisation; technical writing - help documents, user instructions, specifications; * Personal and interpersonal skills including customer service, leadership, teamwork, negotiating, self-management, social and multicultural awareness, relationship and conflict management | | | |
| **Assignment Checklist:** | | |  |  | | --- | --- | | **Requirement** | **Completed** | | Database | [Symbol][Symbol] | | User interface | [Symbol][Symbol] | | Functionality | [Symbol][Symbol] | | Coding | [Symbol][Symbol] | | Testing | [Symbol][Symbol] | | |

**Disclaimer of Plagiarism and Collusion**

I declare that, to the best of my knowledge, this assessment is my own work, and has not been copied from any other student's work or from any other source.

Your Name: AMARBIR SINGH

Enter your name here to indicate you agree to the above statement.

# Final Major Project

These tasks are designed to produce evidence for the assessment of Unit Standard:

## Part 1: Project Proposal

Step 1: Determine a suitable mobile app or web app or a combination of both

The project should take you approximately 20 days effort to complete. You will need to discuss the scope of the project with your tutor before submitting your project proposal.

It is important that you undertake a project that is achievable in the time allowed, and this may result in you undertaking a reasonably self-contained working subset of the total requirements.

The project is expected to include:

* systems analysis - an investigation of the requirements of the target system with an emphasis on user needs
* systems design - an overall coverage of the way the requirements will be met by a computer-based system – it is expected to be a single-user system with **no** dependency on data-communication for the primary facility
* programming - the nominated functionality completed as a working system
* documentation – minimal reference for the user

Step 2***:*** Produce a project proposal. The project proposal must contain:

Project Goal – describe in a single sentence what you will have when you are finished.

Project Objectives – list the objectives of the project – what are the reasons for doing it, what is the intended product designed to achieve. For example, in designing a system that automates a manual process, objectives might include reducing time taken on the process, reducing human error, etc. Also include what you intend to gain from the exercise.

Project Scope – define exactly what you are going to produce. This may be produced as a list, a diagram, storyboard or any other appropriate medium. This section is very important, as it will determine whether you have completed the project or not. If you can define exactly what you are producing at the beginning, it will make completing the project and communicating results much easier. Points to consider:

1. Context - e.g. a satellite monitoring facility to be used by Radio Amateurs
2. Target language platform
3. Stakeholders of the product (ie who will be the end users)
4. What you will **NOT** be doing (ie what is out of scope)

Project Deliverables – List exactly what you intend to include in the final package

For example:

1. Analysis documentation - what needs have to be met by the system
2. Design documentation - how these needs are to be satisfied
3. Executable program - set of programs
4. User documentation
5. Quality plan - how you intend to ensure your user(s) approve of the quality of the final result.
6. Executive Summary of the project - including host hardware and software requirements

* **no** investigation is expected of hardware alternatives, hardware costing, necessary accompanying software purchases, etc.
* any consideration of fees, project costing, etc. are excluded
* arrangements for ownership of the code may be included if relevant

1. Presentation of your project to the class
2. Constructive critical analysis of key learning from the exercise

Resources

List what you will require to complete the project. Show how the project requirements will be met, including allocation of class time and own time

Include details of hardware, software, user input, yourself, mentoring you hope to enlist.

ProjectMilestones

List the milestones of the project and the expected date that they will be achieved. You will need a milestone for the completion date of each deliverable, but there may also be others along the way that require measuring. For example “Completion of user interviews”, which is part of the analysis documentation deliverable.

ProjectSchedule

List all of the activities you will need to do to complete the deliverables, along with expected time needed to perform them and dates they are due by.

Examples of activities you may want to include are:

1. Completion of initial investigation
2. Design Drafted
3. Design signed-off
4. Prototype program completed
5. Stages of program refinement signed-off
6. Program completed
7. Program tested
8. System documentation completed
9. User acceptance completed
10. User report on system acceptability completed

ProjectAssumptions

List the assumptions you have made while producing the plan. For example “that users are available during the project to provide information” or “that software is available at the client site”.

# Part 2 – Complete the project

Undertake your project as per the plan.

Keep a project journal that shows how much time you have spent on the project, what you have achieved, any issues you found, major highs and lows, lessons learned etc. You will need to hand this in for Part 3, so the more you write now, the easier the last stage will be!

Arrange a weekly meeting with your tutor and the project sponsor (client).

Prepare an agenda for those meetings that includes checking progress, review of issues, possible issues (risks) and tasks coming up. Complete minutes and action items following those meetings.

Prepare a simple report for each meeting that shows how you are doing against the project plan, and whether the plan needs updating to reflect any changes you have agreed with your tutor and sponsor.

# Part 3: Review your project

Step 1: Produce a post project review report on your project. The report is to be written formally, and must include:

1. Final Product/Outcome - a description of what you produced
2. Product quality – what you delivered and what you expected to deliver, and the reasons for any variation.
3. How it measured against objectives and measures of success
4. An evaluation of how well you thought the project went – what were the major successes, weaknesses and lessons learned.
5. Future scope
6. Your project journal as an attachment

Step 2: Prepare a 10 minute presentation based on your report and deliver to your class. You may quickly demonstrate your product if time allows, but it is not essential to do so. It is more important that you deliver the bulk of your report.

**Marking**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **% of Grade** | **Excellent 100%** | **Adequate 80%** | **Poor 60%** | **Not Met 0%** |
|
| **Program Specifications / Correctness** | | |  |  |
| 50% | No errors, program always works correctly and meets the specification. | Minor details of the program specification are violated, program functions incorrectly for some inputs. | Significant details of the specification are violated, program often exhibits incorrect behavior. | Program only functions correctly in very limited cases or not at all. |
| **Mark** | 50 | 40 | 30 | 0 |
| **Readability** | | |  |  |
| 20% | No errors, code is clean, understandable, and well-organized. | Minor issues with layout, variable naming, or general organization. | At least one major issue with layout, variable names, or organization. | Major problems with at three or four of the readability subcategories. |
| **Mark** | 20 | 16 | 12 | 0 |
| **Documentation** | | |  |  |
| 20% | No errors, code is well-commented. | One or two places that could benefit from comments are missing them **or** the code is *overly* commented. | Complicated lines or sections of code uncommented or lacking meaningful comments. | No comments present. |
| **Mark** | 20 | 16 | 12 | 0 |
| **Code Efficiency** | | |  |  |
| 5% | No errors, code uses the best approach in every case. | *N/A* | Code uses poorly-chosen approaches in at least one place. | Many things in the code could have been accomplished in an easier, faster, or otherwise better fashion. |
| **Mark** | 5 | 4 | 3 | 0 |
| **Assignment Specifications** | | |  |  |
| 5% | No errors | *N/A* | Minor details of the assignment specification are violated, such as files named incorrectly or extra instructions slightly misunderstood. | Significant details of the specification are violated, such as extra instructions ignored or entirely misunderstood. |
| **Mark** | 5 | 4 | 3 | 0 |