**Diploma in Software and Design**

**Assignment Cover Sheet**

|  |  |  |
| --- | --- | --- |
| **Student’s name:** Rahul Mann | | |
| **Module Name /or number: Mobile App Dev (25 credits)** | | |
| **Assignment title and/or number**:  DSED-03 Android Hangman App | | |
| **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**: | | **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 below:  <https://github.com/xq8787> |
| **Learning Outcomes:** | * Applying appropriate business process modelling tools to analyse and document business processes; * User experience (Ux) design including user interface (UI), HCI principles, and universal accessibility; * Creating accurate and clear technical and user documentation; * Coding – object oriented, procedural; * Facility in multiple common programming languages and integrated development environments (IDEs), which fosters the ability to migrate to new languages, tools and systems; * 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; * Testing on a range of platforms e.g. multiple devices and environments; * Executing tests using manual and automated software testing, and documenting results; * Debugging, which includes debugging utilities, managing bug reports and issue tracking. * 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 * 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]. | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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.

Rahul Mann

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

# DSED-03 Android Hangman App

## Instructions:

* The assessment is an open book exercise – students may consult with others, but finally must present their own work.
* This assessment can be done as a group project, however the complexity of the project must reflect the added members.

### Learning Outcomes

This assessment tests student’s skill in designing basic Android apps.

You can design a Hangman App, **OR** create your own application of a similar level of complexity.

Students will:

1.1 Construct an offline Database to hold data with CRUD

1.2 Create Class and Data modules to hold database calls and common calls

1.3 Create multiple Interfaces

Students may:

2.1 Create a widget

2.2 Download data from the internet

### Hangman

**Requirement**

Design a Hangman app that contains an SQLlite database.

**Functionality Required**

1. Create a profile for the user
2. List the high scores
3. Simulate the game

#### UI Design

Have a look at some of the android apps on play store.

 



#### Basic Functionality Requirements

1. The app should simulate the hangman game.
2. You will need to have separate buttons for all the alphabets, and disable them when they have been selected.
3. Set up a scoring system (based on word difficulty or timing etc.)

#### Submission required

1. Your Android project folder
2. SQlite database file or array of words
3. Database Design(if used)

### Marking Schedule

|  |  |  |
| --- | --- | --- |
|  | Form features | |
|  | 1.1 | Create Class and Data modules to hold database calls and common calls |
|  | 1.2 | Create multiple Interfaces |
|  | 1.3 | Create a profile for the user |
|  | 1.4 | List the high scores |
|  | 1.5 | Simulate the game |
|  | 1.6 | Separate buttons for all the alphabets, and disable them when they have been selected |
|  | 1.7 | Set up a scoring system |
|  | Database Operations | |
|  | 2.1 | Construct an offline Database to hold data with CRUD |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **% 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 |