# Assessment Task 3: Build an

intermediate OOP application

|  |  |
| --- | --- |
| Course code  and title | **ICT50220 Diploma of Information Technology** |
| Unit code and  title | **ICTPRG549 Apply intermediate object-oriented language skills** |
| Due date | 28/05/2024 |
| Resources required | Computer Access with Microphone & Video (optional) & Internet Melbourne Polytechnic lab PC that connected to the internet |
| Decision making rules | To achieve an overall satisfactory result for this assessment task:   * Learners must achieve a satisfactory result for each item in the Observation and/or Assessment Checklist/s |
| Learner instructions | You are required to create and intermediate object-oriented application. There are 2 parts to this task:  **Part A** – respond to a series of questions to create your application.  **Part B** - demonstrate the functionality of your application to your Assessor.  For this task you will:   * Complete it individually. * Write answers to all questions. * Demonstrate your application to your Assessor. * Complete it in class as indicated by your Assessor and submit it by the due date. * Have time to read and review the assessment task in class. * Submit your answers electronically via Moodle, (or in hard copy to your assessor, including the signed cover sheet and   Learner declaration)   * You must answer electronically and save the document as Assessment Task 3 StudentID.docx * You must agree (by clicking on the ‘I confirm radio button) with the assessment submission terms and conditions in Melbourne Polytechnic Moodle prior to the submission   If you have any questions about the task or concerns about your ability to complete the task, please discuss this with your Assessor. |

## Task details

Scenario

Melbourne Polytechnic (MP), formerly NMIT, is a [vocational education](https://en.wikipedia.org/wiki/Vocational_education) ([TAFE)](https://en.wikipedia.org/wiki/TAFE) and [higher](https://en.wikipedia.org/wiki/Higher_education) [education](https://en.wikipedia.org/wiki/Higher_education) institute located in [Melbourne.](https://en.wikipedia.org/wiki/Melbourne) It is predominantly in the northern suburbs with 7 campuses and 4 training sites across Victoria.

|  |  |
| --- | --- |
| **Campuses**  Preston Collingwood Epping Heidelberg Prahran  Greensborough | **Training Sites**  Broadmeadows Eden Park  Yan Yean Ararat |

MP needs to build an intermediate student car park application to activate a student’s parking session through one computer, to be deployed in all parking areas.

System requirement.

* Easy and interactive GUI
* Data required: student name, student ID and car registration number
* Database required at this stage
* Functions should be used
* Sorting by student ID
* Search by student ID
* Contract duration: 3 weeks
* Final project should be run as EXE file.

[YOU SHOULD USE YOUR INITIALS AT THE BEGINNING OF EACH VARIABLE](https://www.google.com/search?rlz=1C1CHBD_en-GBAU856AU856&sxsrf=ALeKk00q5GW_WoRCZ26o5t94Dls_ARuSxg%3A1621762887516&q=%E2%80%A2%2BYOU%2BSHOULD%2BUSE%2BYOUR%2BINITIALS%2BAT%2BTHE%2BBEGINNING%2BOF%2BEACH%2BVARIABLE&spell=1&sa=X&ved=2ahUKEwidvMfGwd_wAhVozDgGHV3lAdsQkeECKAB6BAgBEDE)

i.e. my name is Hatim Mansor, so variables should be: HM\_FirstName

HM\_LastName HM\_Counter HM\_i

### Part A - Development

Answer the following questions and develop the application to the specifications provided in the scenario.

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **Design a database structure that will address all requirements from the scenario and provide a screenshot of the database design.** | | |
|  | SCREENSHOT |  SATISFACTORY |  NOT SATISFACTORY |
|  | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **2** | **Create a text file to create a data base table/file that includes all the required fields as specified in the scenario.**  **Provide a screenshot of the text file.** | | |
|  | SCREENSHOT |  SATISFACTORY |  NOT SATISFACTORY |
|  | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **3** | **In your program, write the required code to connect to the database server.**  **Provide a screenshot of the code.** | | |
|  | SCREENSHOT |  SATISFACTORY |  NOT SATISFACTORY |
|  | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **4** | **Design and implement at least one function that will use Python facilities to handle the database sorting.**  **Provide a screenshot.** | | |
|  | SCREENSHOT |  SATISFACTORY |  NOT SATISFACTORY |
|  | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **5** | **Provide a screenshot for your main interactive GUI user interface** | | |
|  | SCREENSHOT |  SATISFACTORY |  NOT SATISFACTORY |
|  | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **6** | **Design and implement a test scenario to test name, age and registration number.**  **Provide a screenshot of the test table below.** | | | | | | |
|  | ANSWER |  |  |  SATISFACTORY | |  NOT SATISFACTORY | |
| **Priority** | | **Requirement** | **Module** | **Submodule** | **Contents** | | **Expected**  **Results** |
| A | | R87 | MP Student Car Parking Management System | Register a Car | Fill in student name.  Fill in student ID.  Fill in the license plate. | | Enter student name successfully.  Enter student ID successfully.  License plate entered successfully. |
| Fill in student name    Fill in student ID    Fill in the license plate    registration success | | | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **7** | **Produce a code to satisfy all of MP’s requirements. Make sure you have tested the code thoroughly to comply with MP specifications.**  **Copy and paste your code below** | | |
|  | ANSWER |  SATISFACTORY |  NOT SATISFACTORY |
| # Import the required libraries import re import tkinter import tkinter.messagebox import pymysql   # Locomotive number verification def check\_rego\_number(wyz\_rego\_number):  if re.match("^c[0-9]{7}$", wyz\_rego\_number, re.ASCII):  return True  else:  return False  # Vehicle category class Car:  def \_\_init\_\_(self, wyz\_rego\_number, wyz\_std\_obj):  try:  assert check\_rego\_number(wyz\_rego\_number)  self.wyz\_rego\_number = wyz\_rego\_number  self.wyz\_owner = wyz\_std\_obj  except AssertionError:  tkinter.messagebox.showerror("错误", "车辆注册码有误")  raise   # Student ID detection def check\_std\_id(wyz\_stud\_id):  if re.match("^s[0-9]{7}$", wyz\_stud\_id, re.ASCII):  return True  else:  return False  # Student category class Student:  def \_\_init\_\_(self, wyz\_std\_name, wyz\_stud\_id):  try:  self.wyz\_std\_name = wyz\_std\_name  assert check\_std\_id(wyz\_stud\_id)  self.wyz\_stud\_id = wyz\_stud\_id  except AssertionError:  tkinter.messagebox.showerror("错误", "学号格式有误")  raise     # Vehicle management class CarManager:  def \_\_init\_\_(self):  self.wyz\_car\_list = []   def c\_register(self, wyz\_c\_obj):  for car in self.wyz\_car\_list:  if car.wyz\_rego\_number == wyz\_c\_obj.wyz\_rego\_number:  tkinter.messagebox.showerror("错误", "车辆已被注册！")  return False  else:  self.wyz\_car\_list.append(wyz\_c\_obj)  return True   # Student Management class StudentManager:  def \_\_init\_\_(self):  self.wyz\_stud\_list = []   def std\_register(self, wyz\_std\_obj):  for s in self.wyz\_stud\_list:  if s.wyz\_stud\_id == wyz\_std\_obj.wyz\_stud\_id:  tkinter.messagebox.showerror("错误", "学生已存在")  return False  else:  self.wyz\_stud\_list.append(wyz\_std\_obj)  return True   # Search by student ID def search\_studByID(wyz\_stud\_id):  wyz\_connect = pymysql.connect(host="localhost", user="root", password="", database="wyz\_mp\_park",  charset="utf8")  wyz\_cursor = wyz\_connect.cursor(cursor=pymysql.cursors.DictCursor)  wyz\_sql = "select \* from `wyz\_student` where wyz\_stud\_id = %s"  wyz\_cursor.execute(wyz\_sql, [wyz\_stud\_id])  wyz\_find = wyz\_cursor.fetchone()  if wyz\_find is None:   return False  else:  return wyz\_find     # Sort by student ID def sort\_id(wyz\_stud\_list, reverse=False):  wyz\_stud\_list.sort(key=lambda x: x.get("wyz\_stud\_id"), reverse=reverse)  return wyz\_stud\_list     # main database connection and student data query wyz\_connect = pymysql.connect(host="localhost", user="root", password="", database="wyz\_mp\_park", charset="utf8") wyz\_cursor = wyz\_connect.cursor(cursor=pymysql.cursors.DictCursor) wyz\_sql = "select \* from `wyz\_student`" wyz\_cursor.execute(wyz\_sql) wyz\_students = wyz\_cursor.fetchall()  # Registered student information sorted by student ID print(sort\_id(wyz\_students))   # Register to respond to events def register():  wyz\_connect = pymysql.connect(host="localhost", user="root", password="", database="wyz\_mp\_park",  charset="utf8")  wyz\_cursor = wyz\_connect.cursor(cursor=pymysql.cursors.DictCursor)   wyz\_name = wyz\_e1.get()  wyz\_id = wyz\_e2.get()  wyz\_rego\_number = wyz\_e3.get()    if wyz\_name == '' or wyz\_id == '' or wyz\_rego\_number == '':  tkinter.messagebox.showerror("error", "All information cannot be empty")  return False   if not re.match("^s[0-9]{7}$", wyz\_id, re.ASCII):  tkinter.messagebox.showerror("error", "Student number does not meet the requirements")  return False   if not re.match("^c[0-9]{7}$", wyz\_rego\_number, re.ASCII):  tkinter.messagebox.showerror("error", "The vehicle number does not meet the requirements")  return False    wyz\_find\_student = search\_studByID(wyz\_id)  if wyz\_find\_student and wyz\_find\_student.get("wyz\_std\_name") == wyz\_name:  wyz\_sql = "select \* from `wyz\_car` where wyz\_rego\_number = %s"  wyz\_cursor.execute(wyz\_sql, [wyz\_rego\_number])  wyz\_find\_car = wyz\_cursor.fetchone()  if wyz\_find\_car is None:  wyz\_sql = "insert into `wyz\_car` values (%s, %s)"  wyz\_cursor.execute(wyz\_sql, [wyz\_rego\_number, wyz\_find\_student.get("wyz\_stud\_id")])  wyz\_connect.commit()  tkinter.messagebox.showinfo("registration success", "Vehicle registration successful")  return True  else:  tkinter.messagebox.showerror("error", "The vehicle has been registered!")  return False   elif wyz\_find\_student and wyz\_find\_student.get("wyz\_std\_name") != wyz\_name:  tkinter.messagebox.showerror("error", "Student number does not match student name")  return False  else:  wyz\_sql = "select \* from `wyz\_car` where wyz\_rego\_number = %s"  wyz\_cursor.execute(wyz\_sql, [wyz\_rego\_number])  wyz\_find\_car = wyz\_cursor.fetchone()  if wyz\_find\_car is None:  wyz\_sql = "insert into `wyz\_student` values (%s, %s)"  wyz\_cursor.execute(wyz\_sql, [wyz\_id, wyz\_name])  wyz\_sql = "insert into `wyz\_car` values (%s, %s)"  wyz\_cursor.execute(wyz\_sql, [wyz\_rego\_number, wyz\_id])  wyz\_connect.commit()  tkinter.messagebox.showinfo("registration success", "Student registration successful")  tkinter.messagebox.showinfo("registration success", "Vehicle registration successful")  return True  else:  tkinter.messagebox.showerror("error", "Vehicle has been registered")  return False   # GUI graphical interface design wyz\_root = tkinter.Tk() wyz\_root.title("wyz\_Car\_Registration") wyz\_root.resizable(width=False, height=False)  wyz\_lb1 = tkinter.Label(wyz\_root, text="wyz\_Car\_Registration", bg="gainsboro", fg="steelblue", font=("Bold", 30)) wyz\_lb1.grid(row=0, column=0, columnspan=4, pady=10, ipadx=10, ipady=10)  wyz\_lb2 = tkinter.Label(wyz\_root, text="wyz\_Student Name", bg="goldenrod", fg="mediumvioletred", font=("Bold", 15)) wyz\_lb2.grid(row=1, column=1, padx=10, pady=10, ipadx=5, ipady=5) wyz\_e1 = tkinter.Entry(wyz\_root, font=("normal", 13)) wyz\_e1.grid(row=1, column=2, pady=10, ipadx=5, ipady=5)  wyz\_lb3 = tkinter.Label(wyz\_root, text="wyz\_Student ID", bg="goldenrod", fg="mediumvioletred", font=("Bold", 15)) wyz\_lb3.grid(row=2, column=1, padx=10, pady=10, ipadx=5, ipady=5) wyz\_e2 = tkinter.Entry(wyz\_root, font=("normal", 13)) wyz\_e2.grid(row=2, column=2, pady=10, ipadx=5, ipady=5)  wyz\_lb4 = tkinter.Label(wyz\_root, text="wyz\_Rego Number", bg="goldenrod", fg="mediumvioletred", font=("Bold", 15)) wyz\_lb4.grid(row=3, column=1, padx=10, pady=10, ipadx=5, ipady=5) wyz\_e3 = tkinter.Entry(wyz\_root, font=("normal", 13)) wyz\_e3.grid(row=3, column=2, pady=10, ipadx=5, ipady=5)  wyz\_btn = tkinter.Button(wyz\_root, text="wyz\_Register", command=register, bg="orangered", fg="white", font=("normal", 15)) wyz\_btn.grid(row=4, column=1, columnspan=2, pady=10, ipadx=5, ipady=5) wyz\_root.mainloop() | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **8** | **Generate the final binary executable file.**  **Take a screenshot** | | |
|  | SCREENSHOT |  SATISFACTORY |  NOT SATISFACTORY |
|  | | | |

### Part B – Demonstration

You must now explain and demonstrate your application to your Assessor. Use your answers to the questions in Part A to assist you.

You will be assessed on the items in the observation checklist. Ensure to:

* explain the database structure and table you have designed.
* demonstrate the code to connect to the database structure.
* demonstrate the Python facility for sorting.
* display the GUI with relevant fields.
* demonstrate the test scenario.
* display the final code and binary file.

## Observation Checklist: Build an intermediate OOP application

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Learner name** | |  | **Student ID** | |  | |
| **Assessor name** | | Mr. YU | **Date** | | 28/05/2023 | |
| OBSERVATION CHECKLIST  ASSESSOR TO COMPLETE THE FOLLOWING | | | | | | |
| **THE LEARNER:** | | | | **SATISFACTORY** | | **NOT SATISFACTORY** |
| 1 | Correctly explained the database structure they have designed and structure meets MP specifications. | | | ☑ | | ☐ |
| 2 | Correctly explained the database table they have designed. | | | ☑ | | ☐ |
| 3 | Demonstrated the code to connect to the database server. | | | ☑ | | ☐ |
| 4 | Demonstrated use of Python facility to manage sorting of data. | | | ☑ | | ☐ |
| 5 | Displayed the GUI with name, registration and student ID. | | | ☑ | | ☐ |
| 6 | Demonstrated the test scenario. | | | ☑ | | ☐ |
| 7 | Displayed correct code and demonstrated application functions to MP specifications. | | | ☑ | | ☐ |
| 8 | Correctly generated a binary executable file. | | | ☑ | | ☐ |
| **Feedback -** Assessor must include feedback. | | | | | | |
| The candidate has completed all aspect as required above, and basically meet the requirements. | | | | | | |

## Assessment Checklist: Task 3 - Build an intermediate OOP application.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Learner name** | |  | **Learner ID** | |  | |
| **Assessor name** | | Mr. YU | **Date** | | 28/05/2023 | |
| ASSESSMENT CHECKLIST  ASSESSOR TO COMPLETE THE FOLLOWING  THE LEARNER: | | | | | | |
|  | | | | **SATISFACTORY** | | **NOT SATISFACTORY** |
| 1 | Correctly completed all questions in Part A. | | | ☑ | | ☐ |
| 2 | Explained and demonstrated all items in Observation  Checklist. | | | ☑ | | ☐ |
| **Feedback -** Assessor must include feedback about the observed performance. | | | | | | |
| The candidate has complete all questions in Part A and has demonstrated all items in the observation checklist. | | | | | | |

## Assessment Task Summary: Task 3 - Build an intermediate OOP application.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TRAINER/ASSESSOR TO COMPLETE THE FOLLOWING:  **THE LEARNER:** | | | | | | YES | NO |
| 1. | Satisfactorily completed all the questions | | | | | ☐ | ☐ |
| 2 | Satisfactorily completed all items in Observation Checklist | | | | | ☐ | ☐ |
| FEEDBACK **-** Assessor must include feedback | | | | | | | |
|  | | | | | | | |
| OVERALL TASK RESULT | | | | | | | |
| * Satisfactory * Not Satisfactory (resubmission required) – Due date: | | | | | | | |
| DATE ASSESSMENT RETURNED | | |  | | | | |
| TRAINER/ASSESSOR NAME | | |  | | | | |
| TRAINER/ASSESSOR SIGNATURE | | | X | | | | |
| **LEARNER DECLARATION**: Please read, tick and sign below | | | | | | | |
| * I, have been advised of the outcome of this assessment task.   PRINT NAME | | | | | | | |
| LEARNER SIGNATURE | | X | | DATE |  | | |