SINGAPORE UNIVERSITY OF SOCIAL SCIENCES

ICT239

End-of-Course Assessment – January Semester 2022

Web Application Development

INSTRUCTIONS TO STUDENTS:

1. This End-of-Course Assessment paper contains **THREE (3)** questions and comprises **NINE (9)** pages (including the cover page).

2. You are to include the following particulars in your submission: Course Code, Title of the ECA, SUSS PI No., Your Name, and Submission Date.

3. Late submission will be subjected to the marks deduction scheme. Please refer to the Student Handbook for details.

IMPORTANT NOTE

ECA Submission Deadline: Sunday, 22 May 2022, 12 noon.

Please Read This Information before You Start Working on your ECA

This ECA carries 70% of the course marks and is a compulsory component. It is to be done individually and not collaboratively with other students.

Submission

You are to submit the ECA assignment in exactly the same manner as your tutor-marked assignments (TMA), i.e. using Canvas. Submission in any other manner like hardcopy or any other means will not be accepted.

Electronic transmission is not immediate. It is possible that the network traffic may be particularly heavy on the cut-off date and connections to the system cannot be guaranteed. Hence, you are advised to submit your assignment the day before the cut-off date in order to make sure that the submission is accepted and in good time.

Once you have submitted your ECA assignment, the status is displayed on the computer screen. You will only receive a successful assignment submission message if you had applied for the e-mail notification option.

ECA Marks Deduction Scheme

Please note the following:

a) Submission Cut-off Time – Unless otherwise advised, the cut-off time for ECA submission will be at ***12:00 noon*** *on the day of the deadline. All submission timings will be based on the time recorded by Canvas.*

b) Start Time for Deduction – Students are given a grace period of 12hours. Hence calculation of late submissions of ECAs will begin at ***00:00 hrs*** *the following day (this applies even if it is a holiday or weekend) after the deadline.*

c) How the Scheme Works – From 00:00 hrs the following day after the deadline, ***10 marks*** *will be deducted for each* ***24-hour block****. Submissions that are subject to more than 50 marks deduction will be assigned* ***zero mark****. For examples on how the scheme works, please refer to Section 5.2 Para 1.7.3 of the Student Handbook.*

Any extra files, missing appendices or corrections received after the cut-off date will also not be considered in the grading of your ECA assignment.

Plagiarism and Collusion

Plagiarism and collusion are forms of cheating and are not acceptable in any form of a student's work, including this ECA assignment. You can avoid plagiarism by giving appropriate references when you use some other people's ideas, words or pictures (including diagrams). Refer to the American Psychological Association (APA) Manual if you need reminding about quoting and referencing. You can avoid collusion by ensuring that your submission is based on your own individual effort.

The electronic submission of your ECA assignment will be screened through a plagiarism detecting software. For more information about plagiarism and cheating, you should refer to the Student Handbook. SUSS takes a tough stance against plagiarism and collusion. Serious cases will normally result in the student being referred to SUSS's Student Disciplinary Group. For other cases, significant marking penalties or expulsion from the course will be imposed.

Answer all questions. (Total 100 marks)

This paper continues the TMA theme on the development Web Application for viewing and booking of staycation packages.

Question 1 (40 marks)

In TMA Q2, you were asked to follow the MVC framework to design and implement the Web Application. In ECA Q1, you are further asked to follow the Object Oriented Programming principles to implement the Model of MVC through the use of MongoEngine library. Given the following file directory structure:

```
/home/user/Projects/ECA
└─ app
    ├── assets/
    │   ├── css/ # directory with css files
    │   ├── img/ # directory with image files
    │   └── js/  # directory with JavaScript and Data files
    ├── templates/ # directory with html files
    ├── __init__.py
    ├── app.py   # incl. codes for upload files (C/MVC)
    ├── auth.py  # incl. codes for register and login (C/MVC)
    ├── book.py  # incl. codes for booking (MC/MVC)
    ├── staycation.py # incl. codes for packages (MC/MVC)
    ├── users.py # incl. codes for users (MC/MVC)
    ├── dashboard.py # incl. codes for display charts (MC/MVC)
    ├── forms.py # inlc. codes for handling forms using WTForm
    ├── requirements.txt # required libraries
    └── venv/ # python environment
```

Consider the following codes that implements the Model using MongoEngine:

In `__init__.py`:

```
from flask import Flask
from flask_mongoengine import MongoEngine, Document
from flask_login import LoginManager

import pymongo

def create_app():
    app = Flask(__name__)
    app.config['MONGODB_SETTINGS'] = {
        'db':'eca',
        'host':'localhost'
    }
    app.static_folder = 'assets'
    db = MongoEngine(app)

    app.config['SECRET_KEY'] = '9OLWxND4o83j4K4iuopO'
    login_manager = LoginManager()
    login_manager.init_app(app)
    login_manager.login_view = 'login'
    return app, db, login_manager

app, db, login_manager = create_app()
```

In `users.py`,

```
from app import db
from flask_login import UserMixin

class User(UserMixin, db.Document):

    meta = {'collection': 'appUsers'}
    email = db.StringField(max_length=30)
    password = db.StringField()
    name = db.StringField()
```

In `staycation.py`,

```
from app import db

class STAYCATION(db.Document):

    meta = {'collection': 'staycation'}
    hotel_name = db.StringField(max_length=30)
    duration = db.IntField()
    unit_cost = db.FloatField()
    image_url = db.StringField(max_length=30)
    description = db.StringField(max_length=500)
```

In `book.py`,

```
from users import User
from staycation import STAYCATION
from app import db

class Booking(db.Document):

    meta = {'collection': 'booking'}
    check_in_date = db.DateTimeField(required=True)
    customer = db.ReferenceField(User)
    package = db.ReferenceField(STAYCATION)
    total_cost = db.FloatField()

    def calculate_total_cost(self):
        self.total_cost = self.package.duration * self.package.unit_cost
```

(a) Produce comments to each of the above python programs that explains what and how the schema of the data model of the application is implemented. Cover all the following keywords or program statements in your comments:

- Those keywords bolded, such as "`app.config[…]`", "`db = MongoEngine(app)`" and "`meta`"
- `class User(db.Document), class STAYCATION(db.Document), class Booking(db.Document), and each of the attributes defined, including total_cost, etc.`
- `db.DataTimeField, db.InteField, db.FloatField, db.ReferenceField,` and `db.StringField`
- `calculate_total_cost`

(10 marks)

(b) Employ the Blueprint and MongoEngine libraries, extend the source codes given in Q1 (a) to design the control routes for the following processing that before the output is stored in the MongoDB designed in Q1 (a); you may use python-like pseudo codes:

(i) For the admin users, to upload the 3 data files and store the content in MongoDB.

(8 marks)

(ii) For users, to book a staycation package and store the booking record in MongoDB.

(8 marks)

(c) Re-develop your TMA solution to construct the application according to the design in Q1(b).

(14 marks)

Question 2 (20 marks)

Question 2 concerns the re-factoring of the html codes that render the views according to the following directory structure under the templates sub-directory and the `forms.py` file:

```
/home/user/Projects/ECA
└─ app
    ├── assets/
    │   ├── css/ # directory with css files
    │   ├── img/ # directory with image files
    │   └── js/  # directory with JavaScript and Data files
    ├── templates/ # directory with html files
    │   ├── __render_field.html # macro to render WTForms form
    │   ├── base.html # the base html file
    │   ├── login.html # that renders the login view
    │   ├── register.html # that renders the register view
    │   ├── packages.html  # that renders the package view
    │   ├── booking.html # that render the booking view
    │   ├── upload.html # that renders the view for login
    │   └── trend_chart.html # that renders the dashboard view
    ├── __init__.py
    ├── ...
    ├── forms.py
    ├── ...
    └── venv/ # python environment
```

where `forms.py` contains the following codes:

```
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, DateTimeField, FloatField
from wtforms.validators import Email, Length, InputRequired

class RegForm(FlaskForm):
    email = StringField('Email',  validators=[InputRequired(), Email(message='Invalid email'),
Length(max=30)])
    password = PasswordField('Password', validators=[InputRequired(), Length(min=5, max=20)])
    name = StringField('Name')
```

and `__render_field.html` contains the following codes:

```
{% macro render_field(field) %}

<div class="form-group">
    <label for="{{ field.name }} "><h3>{{ field.label.text.capitalize() }}</h3></label>
    {{ field(class_='form-control', **kwargs)|safe }}
    <ul>
        {% for error in field.errors %}
        <li style="color:red;">{{ error }}</li>
        {% endfor %}
    </ul>
</div>
{% endmacro %}
```

(a) Provide comments on the above two files to explain how the codes work.

(6 marks)

(b) Refactor and construct the code according to the provided structure below. The refactored html files contain all the source codes necessary for the rendering of all the views in the following 7 files:

- `base.html`
- `login.htm`
- `register.html`
- `packages.html`
- `booking.html`
- `upload.html`
- `trend_chart.html`

(14 marks)

Question 3 (40 marks)

The admin user would like to have further insights into the popularity of the packages. Experiment with the visualization of data so the dashboard view would be added two more links to be clickable in the dashboard view sidebar component:

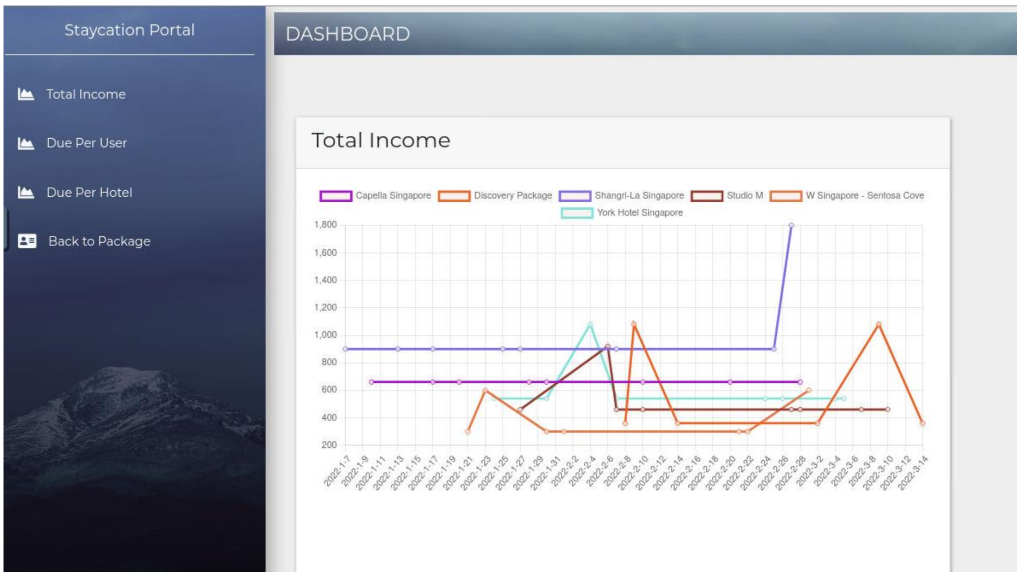

Figure Q3 (a): Due Per User, Due Per Hotel, and Back to Packages links

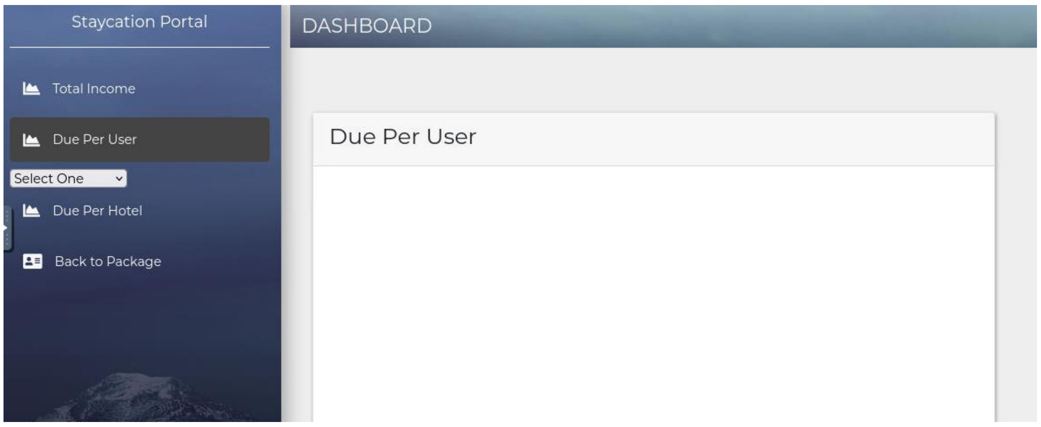

Figure Q3 (b) (i): When Due Per User link is clicked

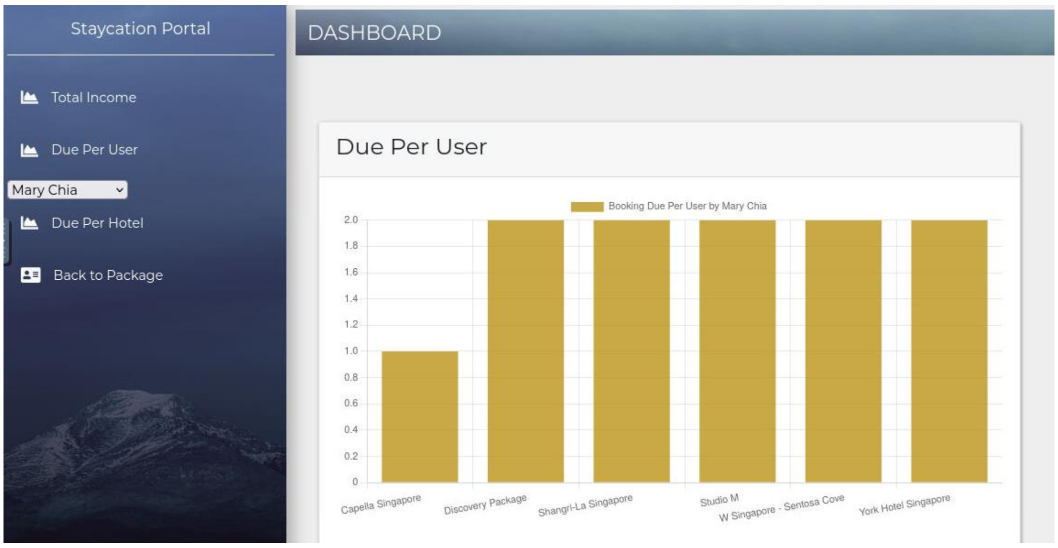

Figure Q3 (b) (ii): When a particular User in Due Per User menu is selected

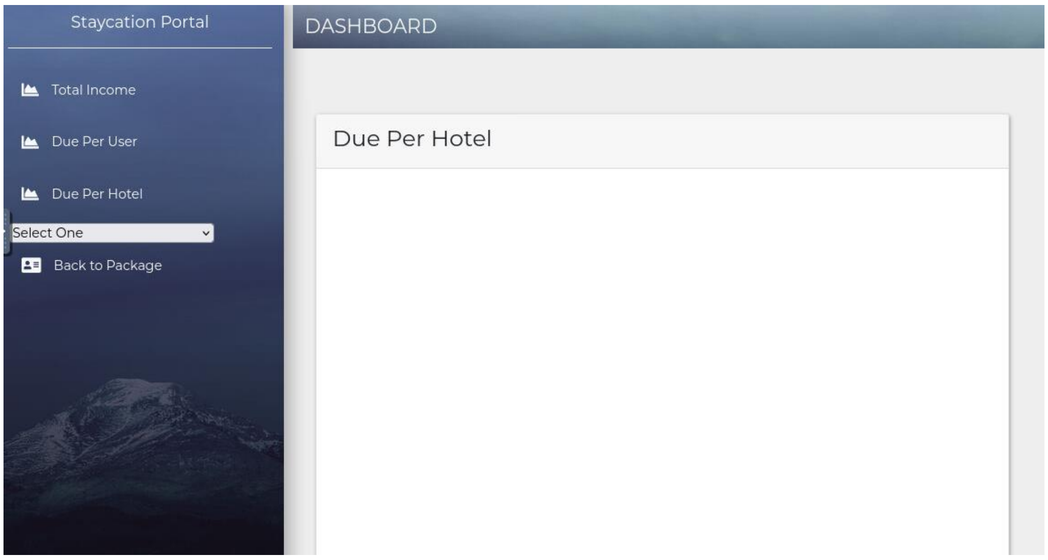

Figure Q3 (c) (i): When Due Per Hotel link is clicked

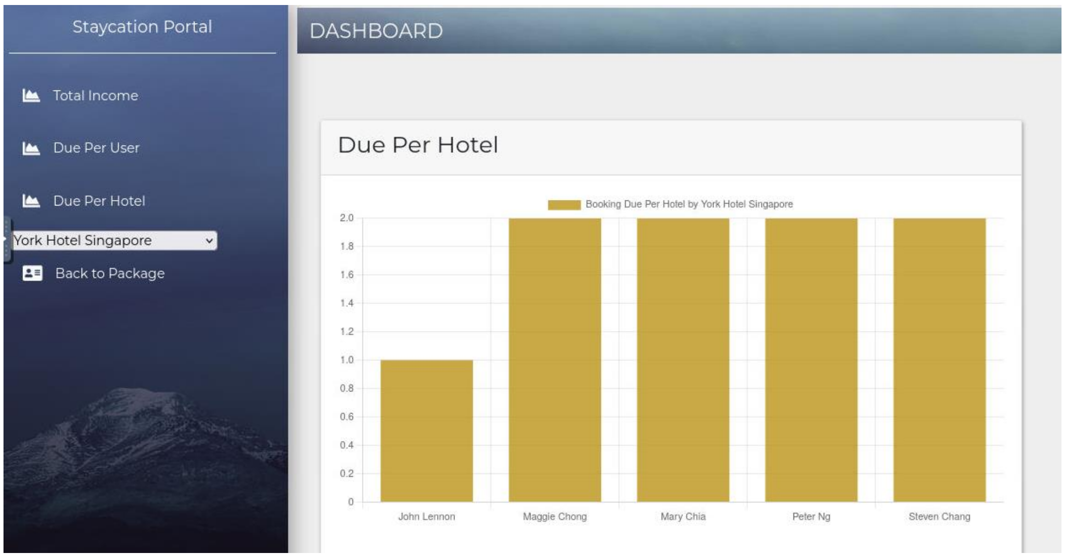

Figure Q3 (c) (ii): When a particular Hotel in Due Per Hotel menu is selected

(a) Design and implement the View and Controller codes that produce the sidebar as required in Figs Q3(a), (b) and (c). The dropdown menu in Figure Q3(b) and (c) would contain all the user names registered and the hotel names of the packages in the website, respectively.

(10 marks)

(b) Apply the data model as required in Q1(a) to implement the chart shown in **Figure Q3 (b)**. The trend data is to be retrieved from the database.

(15 marks)

(c) Apply the data model as required in Q1(a) to implement the chart shown in **Figure Q3 (c)**. The trend data is to be retrieved from the database.

(15 marks)

----- END OF ECA PAPER -----