Student ID: 22036698

UWE Bristol University of the West of England

BSc Computer Science

Module Title

Advanced Software Development

Assessment Title

Individual Reflective Report

Assessment Weighting

10% of the module mark

Student Name

Reece Turner

(09/01/2024)

Student ID: 22036698

Contents

Student ID: 22036698

Introduction

In this report I am reflecting on my personal contributions to the project along with what I have learned throughout the module. Furthermore, I cover a review of different technologies or methods could be used in the software lifecycle.

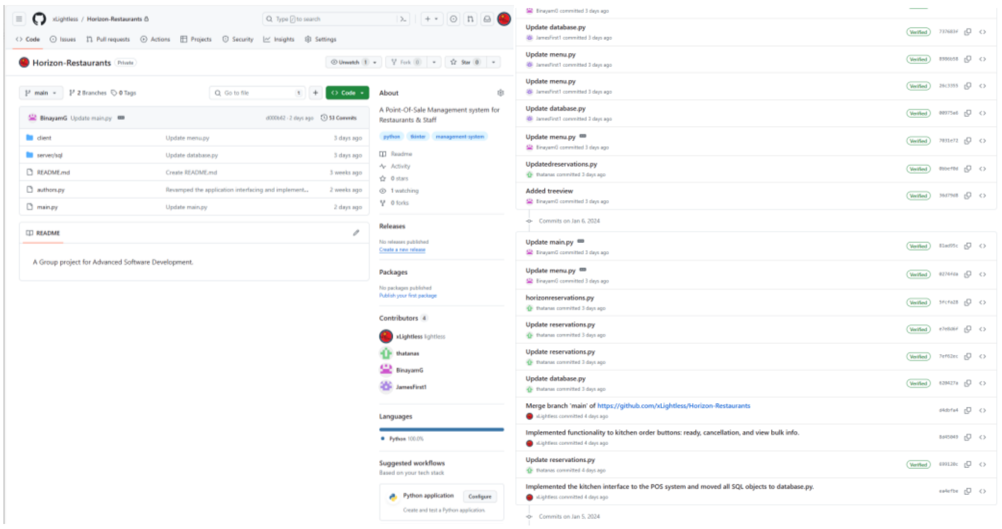
Contribution and challenges

Difficulty Matrix:

Responsibilities	Easy	Intermediate	Difficult
Planning		X	
Database		X	
Login	X		
Non-functional requirements			X
User Management			X
Kitchen/Orders		X	

Regarding the methodological approach, in the aforementioned 'difficulty matrix' I outlined my responsibilities and how challenging each task was. I felt that it was necessary to hit all the requirements and with the lack of engagement from peers I had limited time to make a fully functional application. Communication became dormant and it meant I had to take a leadership role to ensure the portfolio was adequate. As a result, over time, we managed to collectively work together after conversations towards the end to produce a sub-optimal solution.

Learning Reflections and Emerging Technologies



BinayamG Update main.py d000b62 · 2 days ago 53 Commits

client	Update menu.py	3 days ago
server/sql	Update database.py	3 days ago
README.md	Create README.md	3 weeks ago
authors.py	Revamped the application interfacing and implement...	2 weeks ago
main.py	Update main.py	2 days ago

README

A Group project for Advanced Software Development.

A Point-Of-Sale Management system for Restaurants & Staff

python tkinter management-system

No releases published
Create a new release

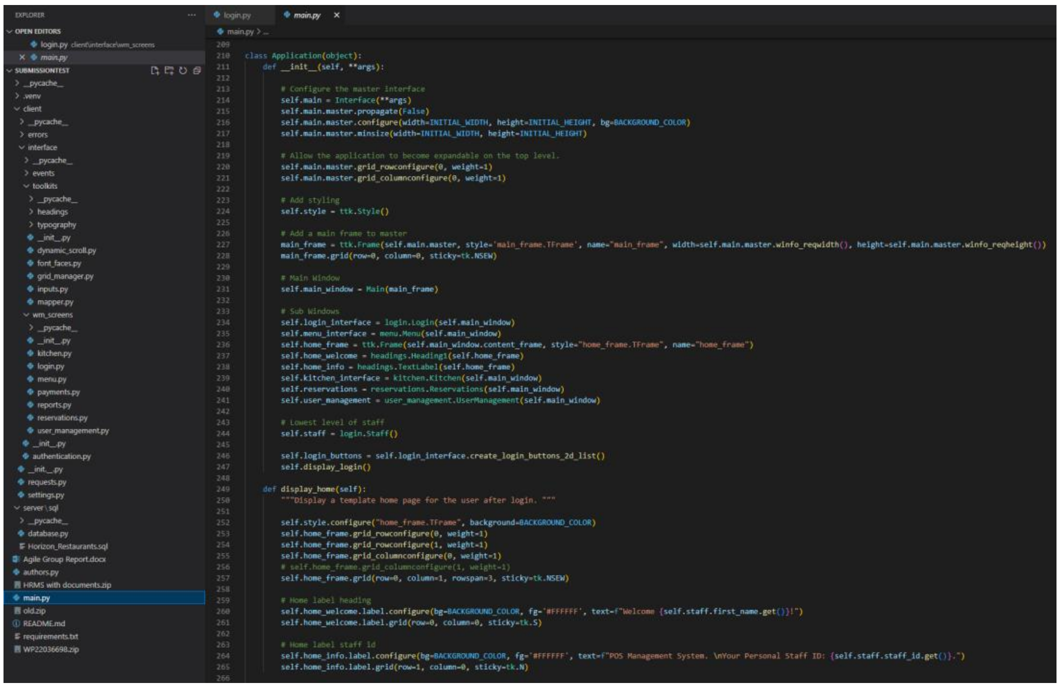
No packages published
Publish your first package

xLightless lightless
thatanas
BinayamG
JamesFirst1

```
BinayamG committed 3 days ago
Update database.py
JamesFirst1 committed 3 days ago                          7376834
Update menu.py
JamesFirst1 committed 3 days ago                          8986b58
Update menu.py
JamesFirst1 committed 3 days ago                          26c3355
Update database.py
JamesFirst1 committed 3 days ago                          00975a6
Update menu.py
BinayamG committed 3 days ago                             7831e72
Updatedreservations.py
thatanas committed 3 days ago                             8bbef8d
Added treeview
BinayamG committed 3 days ago                             36d7968

Commits on Jan 6, 2024

Update main.py
BinayamG committed 3 days ago                             81ad95c
Update menu.py
BinayamG committed 3 days ago                             8274fda
horizonreservations.py
thatanas committed 3 days ago                             5fcfe28
Update reservations.py
thatanas committed 3 days ago                             e7e8d6f
Update reservations.py
thatanas committed 3 days ago                             7ef62ec
Update database.py
thatanas committed 3 days ago                             620427a
Merge branch 'main' of https://github.com/xLightless/Horizon-Restaurants
xLightless committed 4 days ago                           d4dbfa4
Implemented functionality to kitchen order buttons: ready, cancellation, and view bulk info.
xLightless committed 4 days ago                           8d45049
Update reservations.py
thatanas committed 4 days ago                             699128c
Implemented the kitchen interface to the POS system and moved all SQL objects to database.py.
xLightless committed 4 days ago                           ea4efbe

Commits on Jan 5, 2024
```

Fig. 1: GitHub project page and commits.

Reflecting on the learning with emerging technologies in mind, we proceeded by using DevOps in a cloud computing setting. More specifically I am referring to GitHub; a platform that enables developers to work on a repository - asynchronously, if required. With limited time it was the best course of action to ensure we apply CI/CD to streamline the process. GitHub enabled us to push, pull, merge, and publish code from our local machines to either the main branch or other branches influencing each person's actions by taking control over their section of work to create a final product.



```python
class Application(object):
    def __init__(self, **args):

        # Configure the master interface
        self.main = Interface(**args)
        self.main.master.propagate(False)
        self.main.master.configure(width=INITIAL_WIDTH, height=INITIAL_HEIGHT, bg=BACKGROUND_COLOR)
        self.main.master.minsize(width=INITIAL_WIDTH, height=INITIAL_HEIGHT)

        # Allow the application to become expandable on the top level.
        self.main.master.grid_rowconfigure(0, weight=1)
        self.main.master.grid_columnconfigure(0, weight=1)

        # Add styling
        self.style = ttk.Style()

        # Add a main frame to master
        main_frame = ttk.Frame(self.main.master, style='main_frame.TFrame', name="main_frame", width=self.main.master.winfo_reqwidth(), height=self.main.master.winfo_reqheight())
        main_frame.grid(row=0, column=0, sticky=tk.NSEW)

        # Main Window
        self.main_window = Main(main_frame)

        # Sub Windows
        self.login_interface = login.Login(self.main_window)
        self.menu_interface = menu.Menu(self.main_window)
        self.home_frame = ttk.Frame(self.main_window.content_frame, style="home_frame.TFrame", name="home_frame")
        self.home_welcome = headings.Heading1(self.home_frame)
        self.home_info = headings.TextLabel(self.home_frame)
        self.kitchen_interface = kitchen.Kitchen(self.main_window)
        self.reservations = reservations.Reservations(self.main_window)
        self.user_management = user_management.UserManagement(self.main_window)

        # Lowest level of staff
        self.staff = login.Staff()

        self.login_buttons = self.login_interface.create_login_buttons_2d_list()
        self.display_login()

    def display_home(self):
        """Display a template home page for the user after login. """

        self.style.configure("home_frame.TFrame", background=BACKGROUND_COLOR)
        self.home_frame.grid_rowconfigure(0, weight=1)
        self.home_frame.grid_rowconfigure(1, weight=1)
        self.home_frame.grid_columnconfigure(0, weight=1)
        # self.home_frame.grid_columnconfigure(1, weight=1)
        self.home_frame.grid(row=0, column=1, rowspan=3, sticky=tk.NSEW)

        # Home label heading
        self.home_welcome.label.configure(bg=BACKGROUND_COLOR, fg='#FFFFFF', text=f"Welcome {self.staff.first_name.get()}!")
        self.home_welcome.label.grid(row=0, column=0, sticky=tk.S)

        # Home label staff id
        self.home_info.label.configure(bg=BACKGROUND_COLOR, fg='#FFFFFF', text=f"POS Management System. \nYour Personal Staff ID: {self.staff.staff_id.get()}.")
        self.home_info.label.grid(row=1, column=0, sticky=tk.N)
```

Fig. 2: Application main.py code where sub-windows are their own objects. Each object has their own display function.

Moreover, with the implications of using GitHub, we structured our development using SOLID principles. Our code is based on the Single Responsibility Principle meaning that each piece of functionality should have its own object or classification so that we reduce tight coupling as much as possible because this coupling can have a negative effect on Object Oriented Programming. The result produced a top-down application where everything was combined in an application object.

Portfolio Development Justification

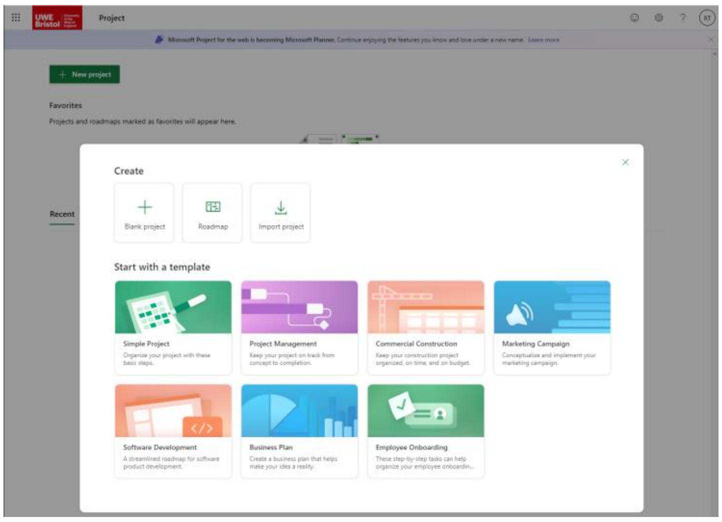
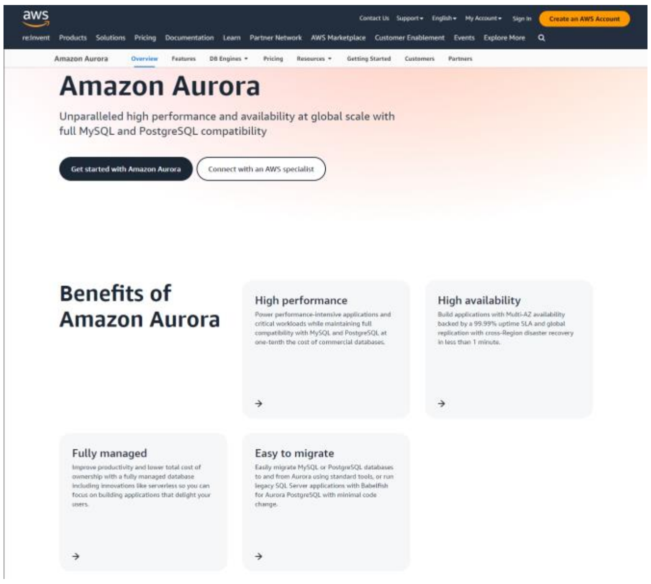


Fig. 3: Microsoft Project document creation page.

In hindsight, our project organisation lacked substance; we attempted to use Trello to map out our tasks but there was lack of structure. If we were to do this again, I would much prefer Microsoft Project due to cloud-based architecture utilisation and its functionalities like Gantt Charts. It would provide our group with milestones and a solid ground for agile SPRINT backlogs.



Amazon Aurora

Unparalleled high performance and availability at global scale with full MySQL and PostgreSQL compatibility

Benefits of Amazon Aurora

High performance

Power performance-intensive applications and critical workloads while maintaining full compatibility with MySQL and PostgreSQL at one-tenth the cost of commercial databases.

High availability

Build applications with Multi-AZ availability backed by a 99.99% uptime SLA and global replication with cross-Region disaster recovery in less than 1 minute.

Fully managed

Improve productivity and lower total cost of ownership with a fully managed database including innovations like serverless so you can focus on building applications that delight your users.

Easy to migrate

Easily migrate MySQL or PostgreSQL databases to and from Aurora using standard tools, or run legacy SQL Server applications with Babelfish for Aurora PostgreSQL with minimal code change.

Fig. 4: Amazon Aurora. Manages DBaaS systems.

Secondly, I would justify that the local method of hosting each person's database deemed ineffective leading to inconsistencies. Future methods like using a Database as a Service (DBaaS) would mean we can host a single Schema in the cloud and make it interdependently accessible and have project compatibility.