Evaluation

Computer Science NEA

21HeffernaJN42

Contents

Evaluation

You need to cover the following, each should have a separate paragraph.

- Effectiveness of the programming language and the tools you have used.
- Compared your system to the commercially available systems in your investigation
- Identified good and weaker aspects of your system, identifying what you would have liked to have done better. You don't actually have to make these improvements so you can be imaginative!!
- Explained your strengths and weaknesses in the design and development of the system
- Identified any personal changes you would make if you were asked to repeat the process to avoid any of the above weaknesses.

For my coursework, I redesigned and improved the fully paper-based system of a restaurant. This meant converting the ordering and reservation booking system into a functioning digital-based system. A lot of thought and planning has gone into the factors used in the design and development of this system, in which I will explain below.

Programming Language

The programming language I have chosen to use for this project is Python. Python is a widespread programming language that is compatible with a large variety of software. As a result of this, I believe that this was a great choice as many of the devices in the restaurant would be able to run my program.

In addition to this, Python has excellent memory management as the program's objects and data structures are stored in a private heap, which is managed by Python's built-in memory manager. This ultimately allowed me to focus on the code and not worry about the memory management aspect.

Additionally, Python is a functional, object-oriented, and procedural programming language. In particular, I enjoyed using the object-oriented style of programming, with the use of objects, classes, etc. The use of OOP allowed me to break down my whole program into sections, which ultimately helped me focus on one section at a time. Also, by breaking down each problem, it was easier to debug and find a fix for my program's errors. Not only this, OOP allowed me to reduce repeating code as I would simply use features such as inheritance, calling functions by reference, etc.

A minute, but appreciated, detail present in Python is the use of dynamic typing. This essentially allows the user to not have to declare variables. This is because the data types of variables are automatically assigned during the runtime of the program. As a result of this, the user will save a lot of time and not have to stress about declaring the variables correctly.

However, Python was not perfect. For example, as Python is an interpreter language, each line of code is executed one by one, which will take a long time, especially if you have a large program. Not only this but due to the dynamic typing, Python has to declare the variables during runtime, which overall places large loads onto the CPU and memory.

Another fault that is caused by the dynamic typing is that runtime errors are expected to occur. For example, variables that have been declared as an integer may try and store a string, which will cause the system to produce an error. Whereas, if the data type of the variable was declared manually, this error could be avoided.

Libraries/Modules

Due to the vast range of libraries available in Python, I was able to utilise a large amount of features I wasn't able to before. Not only did this make my code look much simpler and easy to understand, but it also saved a lot of time. An example of a library I used is the re (Regular Expressions) library. The re library is used to detect whether a string matches the format that is given by the user. This is particularly useful in my program as I used it to validate the fields entered by the user, such as email address.

Another module I used is the DateTime module. As date and time are not data types in Python, it is very difficult to implement them into programs. DateTime allows the user to complete operations on date and time, such as adding or subtracting hours, minutes, or seconds from a certain time, etc.

In addition to this, a critical module that I imported is 'pyodbc'. This module essentially allows Python to make changes to ODBC databases, which is what I have used for my system. This allows the user to create, delete, or alter the database through the use of a simple command.

QT Designer

Having experimented with Tkinter and PyQt in the past, I have decided to use PyQt/QT Designer to design and create the GUIs for my program.

QT Designer is an effective application for creating interfaces due to the efficient yet simple design of the software. All of the tools you can use for your design are clearly displayed on the side, in order for the user to fully utilize the features available to use. This is known as a what-you-see-is-what-you-get (WYSIWYG) user interface.

QT offers the user with a large range of buttons, input widgets, display widgets, layouts and much more. The input widgets have features which produce a signal to the computer as soon as these input fields are used. This is particularly useful as it allows event-driven programming, which is essential for a restaurant booking system. In addition to this, the use of widgets allowed multiple screen to alternate between windows. This is important for any application, as the user wouldn't want a large amount of window to popup in response to a button click.

Converting my UI files into PY files was quite a simple process. It involved using Command Prompt and a Python tool called pyuic5. The tool would essentially create all of the QT Tools I used into Python code. I would then import this Python file into my main code and utilise it from there.

Another advantage that I have found from using QT Designer was the large user base. This allowed me to learn from a variety of sources online, which eventually helped me solve the problems I faced during the development of my project. https://doc.qt.io is a particularly useful website as it shows the user have to fully utilize all of the functions available to each input widget, such as pushbutton.show(), etc.

Integrated Development Environment

My IDE of choice for this project was VS Code. VS Code is an IDE which offers a large range of features such as file explorer, editor window, and much more. VS Code can be used for a variety of programming languages, which can help the programmer explore into various areas in programming.

Using Virtual Studio Code was an excellent experience due to large array of factors. For example, the simplicity of VS Code allows first time users to grasp the features, which is essential for IDEs. In addition to this, VS Code offers a feature called Intelli-Sense. This feature can detect faults in code such as incomplete sections of code, etc. Also, if a user has not declared a variable properly, intelli-sense will automatically declare this variable to avoid any hassle for the user.

Virtual Studio Code also offers a large range of extensions and support. This is very important for programmers as not only can extensions make programming more efficient, but it can also help ignite creativity within the programmer. A particular useful extension is called TODO Highlight. This extension simply allows the user to highlight sections of code that need to be addressed. This helped me identify sections of code easily, while also reminding me to complete certain section of code. The large amount of support available is also very useful as users can watch video guides on how to operate and navigate throughout VS Code.

In addition to this, you can link your repository to VS Code to make importing and saving files easier than ever. In particular, this helps control file management which is essential for large projects.

Database

In order to gain access and alter my database, I have used SQL. Structured Query Language is a widely used database management language as it provides the user the ability to create, alter, and delete tables within a database. Also, SQL allows the feature to create relationships between my tables, which has allowed data extraction to become more efficient. This is done through the ‘INNER JOIN’ feature.

The main advantage of using SQL as my database management language is that the commands are easily understood as they are in English. This essentially means that prior knowledge is not necessary when programming with SQL.

Another advantage is that SQL integrates well with Python. This allowed me to execute queries within my program, instead of having to open the SQL application itself. As a result of this, my coding has become more efficient as more time was saved as all of my code was

in one editor. In addition to this, as SQL was being used in Python, it allowed me to link the query results into my display widgets from QT Designer, such as the importing records into a table widget.

Also, another factor of SQL that I like is the speed it provides the user. As SQL operates at a high-speed, data retrieval is very efficient and fast. This will save the user a lot of time when programming, especially when dealing with large data sets.

Comparison to Desk Based Research

Throughout the development of my project, I have tried to replicate the positives features that I discovered within other systems.

In comparison to Eat App, I believe that I have implemented some features that are important. For example, similarly to Eat, my software is designed for the staff to ensure that all of the reservations are reserved properly, e.g. all details correctly inputted, correct reservation of table, etc. In addition to this, I really liked the simplistic design of the User Interface. I tried to produce a system similar to this, which would it easier for staff to navigate and utilize the features throughout the system. This is essential for systems as it will require less staff training.

However, the majority of useful features that I discovered within the Wisely app was not implemented into my system as I believe that they were useful, but not essential for my system. In particular, I did not add the feature called 'Waitlist'. This feature simply creates a waitlist for the system to automatically fill up cancellations etc. I did not believe it was critical for my system, as the main goal for my system was to create reservations and order food and beverages to tables. In my investigation, I did state that I would complete this as an extension, but I may have overestimated my production capability within the time period.

In the Open Table application, an aspect of the system that I enjoyed was the Calendar Page. The efficient and simplistic design of the page resonated with me, which enabled me to add this onto my system. I imported the feature where the user would simply click on a date in the calendar, then all of the reservations for the day would appear in a table, making it easy for the user to view the reservations for the day. In addition to this, I also integrated the sequential booking process of Open Table. This is where the booking process would follow the same sequence of pages in order to create reservations. I believe that this would be important as it ensures that all of the necessary information is inputted before progressing onto the next page.

Overall, I am satisfied with all of the features that I have integrated due to the desk-based research.

Good Aspects of System

Throughout my system, I believe that I have produced a balanced system, that provides the user with the essential features.

For example, one feature I have added is different access levels. This is seen in the system as only managers have access to the management tab. As a result of this, the system can maintain a level of security due to the fact that only authorised users can make alterations to management details.

Another feature that I added were high levels of validation. This is critical for any system that requires the user to input data as erroneous data could cause the system to malfunction/crash. In order to avoid this, I place validation on every single input field, such as line edit, combo box, etc. However, as I used QWidgets, most widgets already contain some form of validation. For example, in the List Widget, it is impossible for the user to select any value other than the values already listed in the widget. However, for the line edits, I validated each input based on the data they are inputting, such as email having a format of xxxx@xxx.xxx.

In addition to this, I tried to provide the user with a straightforward experience throughout the booking and ordering process. This was mainly completed through the use of a sequential booking process, as users must enter the essential details before proceeding. Also, I tried making the UI as simple as possible by only allowing one task per screen/page. This also provides the user with a straightforward experience as everything is spaced out, making the task easy to see.

Another feature that improves the system is that staff can search for and delete bookings. I like this feature as staff can easily view the reservations for a certain day by simply clicking on the date. This page will display the reservation's party size, customer name, table number, and time. This part is essential for restaurants as it can help them prepare for larger groups, etc. The feature of allowing staff to delete bookings is also important for restaurants as it allows the schedule to be freed up, which can allow further bookings.

Finally, an essential aspect for management that I added onto the system is the flexibility of the menu. This basically means that management have full control of the menu, allowing them to create, alter, or delete menu items. As a result of this, managers can keep the menu fresh with new additions.

Shortcomings of the System

As shortcomings are inevitable when designing a system, here are some shortcomings that I have experienced:

One shortcoming that this system has experience is the lack of encryption. During the design of my system, I proposed as an objective that I would like to add encryption onto my system. However, as I did not prioritise this feature, since it is not critical for the function of the system, I was not able to attempt to add this to my system.

Another shortcoming that I have faced is the absence of the feature to alter reservations. This feature would be desired by staff, but is not essential for the function of the system. The main reason why I was not able to implement this onto my system is that I could not

think of a way to display all of the unavailable tables, without having to go through the original booking process.

In addition to this, a shortcoming that the system has faced is that managers are unable to add new staff members onto the system. Similarly to the alteration of reservations, this feature is desired but not essential. However, I believe that this would be an easy implementation, but I put it on the bottom of my priority list, making me not able to complete the task within the time frame.

Also, another shortcoming is that the system does not include a takeaway section. As stated in my Discussion, I left this feature as an extension, which I was not able to complete.

Finally, a feature that I was not able to add onto the system was the feature to pre-order food/drinks onto a reservation. I wasn't able to implement this into my system as I could not figure out how to complete this task with my current database set-up. By adding this onto my system, the reservation table would have to be linked to the 'menu item' table, instead of linking the food/drink orders to the table they are being sent to.

Improvements of the System

Due to the various shortcomings, there are an array of improvements that could be made onto the system.

For example, I can add encryption onto all of the data in the database. This could be done by encrypting the data based on the value of the primary key of the record. Although the key may not be as complex as others, it would still add encryption onto the data. By doing this, the system will have an increased amount of security, keeping the sensitive information of customers safe.

Another improvement I can add is by storing the ingredients of the menu items in the database. This could be done to find various ingredients that may cause allergies for customers. Not only will this make the system become more professional, but it also increases the safety for the customers.

An improvement that I can add onto the system is the feature to alter reservations. This could be done by opening a new window, which is similar to the booking process pages, but input all of the current reservation details. Once the user has make the reservation, the system will remove the current reservation, and create the altered reservation. Although it is a long process, it may be done with an extension of time.

Another simple improvement that I can add is the feature to add new staff members onto the system. This can be done by simply replicating the 'Alter Staff' GUI but instead of altering an existing staff member when the confirm button is clicked, a new record is created with the details provided by the user. However, similarly to the other management features, only a manager will be able to complete this task.

In addition to this, another aspect that I could improve is the statistics page. Currently, it only sorts the menu items into most selling and least selling. However, I believe that I could

add more statistics such as revenue per day, most customers per day, etc. This could be done through executing the correct SQL queries to obtain the correct values to input into the statistic tables.

Another feature that I failed to add, which could improve the system, is the absence of customer confirmation, such as email confirmation. This could be done using external libraries such as smtplib, ssl. These libraries allow Python to connect to Gmail's SMTP server, which will allow it to send Plain-Text Emails.

Strengths in Approach of Design

The approach of design I used for this project was very similar to the Waterfall Design approach. This meant that each stage was executed sequentially, following a strict order.

The documents that I produced using this approach are:

The Discussion document is the first stage of development. In this document, I outlined the substantial problem the restaurant is facing, while also listing the broad aims and limitations of the solution I was proposing. This essentially helped me develop a goal, while also providing a skeleton to my system.

Following the Discussion document, the Investigation document was produced. In this document, I performed Desk-Based Research in order to gain insight from all systems that have a similar purpose to my solution, mentioning all of the good features that they propose. After this, in order to gain insight from the staff members that are going to be using the new system, I asked them a series of question which are relevant to the design of the system. This helped me overall determine what are the needs and necessities of my proposed solution. Once I did this, I began to investigate the current system of the restaurant, stating the processing methods, limitations, etc. After gaining all of this information, this enabled me to fully create the objectives and success criteria of my system.

Once the Investigation was finished, I began working on my Design document. The Design document is pretty much self-explanatory. In this document, I began to design my system, creating a series of sub-problems that I will need to find solutions to. After this, I designed the UI for the program, creating an interface for each of the pages in my system. I then began to design my database, stating the methods of access, data structures, ERDs, and data dictionary. This ultimately provided me with a strong goal in mind, making the development goal very clear.

After I had completed my Design document, I began coding to create my first Prototype of my system. Once I had done that, I started completing the Prototype document. In the Prototype document, I listed all of the features that were present in my current prototype, including justification for adding those features. After I had done that, I listed all of the features that were absent from my prototype, also justifying why there were not present. Then, I provided the document with screenshots of all of the features present in my prototype. After this, I then evaluated my Prototype, listing the positives and negatives of

the system. Overall, the prototype helped me identify areas in my program which need working on, while also identifying the positive areas of my system.

After identifying all of the negatives/shortcomings in my Prototype, I began amending these shortcomings in my Post-Prototype Refinement of Design. In this document, I described fully all of the problems that were discovered in my prototype. After this, I produced updates to my system that resolved these problems, uploading screenshots of each solution to the document.

After this, I produced a testing table for my system, which lists all of the tests that have undergone, while listing the expected result and actual result. Alongside this, I produced a series of video clips which display the tests being done. This is done to ensure that the system functions correctly, while also ensuring the security of the database, through the use of validation.

Drawbacks in Approach of Design

As my approach to design was very similar to the Waterfall approach, this meant that everything was already planned out and set in stone. As a result of this, once I had stated what I proposed the system to be, it would be very difficult to make alterations to these propositions. This can be problematic as if a better alternative to the solution is found, it would be hard to go back on my previous statements. In order to improve my approach of design, I suggest that the developer should implement an Agile approach in order to increase the flexibility of the development of the program. This will be more beneficial as developers can constantly continue to improve previous sections of code.

Another drawback of this design approach is the delay of the testing phase for the project. As testing was only done once the system itself was completed, unnoticed errors may occur. This will become a problem for the developer as this error could cause a chain reaction throughout the system. As a result of this, the developer may have to request for an extension to the development time in order to fix this issue. In order to fix this issue, I propose that the developer should test the most recent section of code once it has been developed. This should avoid any errors spiralling into other sections of code during development.

Conclusion

In conclusion, I believe that the production of my system was a success as it provided the business with all of the necessities of a restaurant manager/booking system. However, if I were to complete this project again, I would do some things differently. First of all, I would use a more Agile-based approach of design, such as testing throughout the development of each stage. This would ultimately reduce the hassle of discovering large errors at the end of system development. In addition to this, I would like to try and challenge myself by using a different programming language, or even converting the booking system into a website as a whole.

On the other hand, this project has taught me an abundance of lessons. For example, I have learned the importance of validation within a system, as one piece of erroneous data may cause a series of underlying issues. Overall, I have gained a lot of insight from this project and will carry it forward in future projects.