



**PATE’S GRAMMAR SCHOOL**

**COMPUTING DEPARTMENT**

**Unit 3/4 – Programming Project**

CANDIDATE NAME

Sri

EXAM NUMBER

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Table of Contents**

[(1) Analysis of the problem (10 Marks) 3](#_heading=h.gjdgxs)

[(i) problem definition 3](#_heading=h.30j0zll)

[(ii) Stakeholders 5](#_heading=h.1fob9te)

[(iii) RESEARCH THE PROBLEM 6](#_heading=h.2et92p0)

[(iv) SPECIFIY THE PROPOSED SOLUTION 9](#_heading=h.tyjcwt)

[(2) Design OF THE SOLUTION [15 marks] 11](#_heading=h.3dy6vkm)

[(i) DECOMPOSE THE PROBLEM 11](#_heading=h.1t3h5sf)

[(ii) DESCRIBE THE SOLUTION 12](#_heading=h.4d34og8)

[(iii) DESCRIBE THE APPROACH TO TESTING 13](#_heading=h.2s8eyo1)

[(3) DEVELOPING THE SOLUTION (25 mARKS) 14](#_heading=h.17dp8vu)

[(i) INTERATIVE DEVELOPMENT PROCESS 14](#_heading=h.3rdcrjn)

[(ii) TESTING TO INFORM DEVELOPMENT 15](#_heading=h.26in1rg)

[(4) EVALUATION (20 mARKS) 16](#_heading=h.lnxbz9)

[(I) TESTING TO INFORM EVALUATION 16](#_heading=h.35nkun2)

[(II) SUCCESS OF THE SOLUTION 17](#_heading=h.1ksv4uv)

[(III) DESCRIBE THE FINAL PRODUCT 18](#_heading=h.44sinio)

[(IV) MAINTENANCE AND DEVELOPMENT 19](#_heading=h.2jxsxqh)

[APPENDIX A - BIBLIOGRAPHY 20](#_heading=h.z337ya)

# (1) Analysis of the problem (10 Marks)

## (i) problem definition

*(a) Describe and justify the features that make the problem solvable by computational methods.*

*(b) Explain why the problem is amenable to a computational approach.*

Describe your solution. What is it? Why do they need a solution? Why is better than the solution that they currently have or why am I going to emulate a current solution for my clients?

In this section you must talk about why your solution is suited to a computer. You Can’t just say “well it’s a website, so it can’t be done on paper!”. You must compare to non-computer methods i.e. creating a website for Donuts-r-us means that they will be able to store client details, target advertising, have a much wider audience than simply creating a poster.

You could also say things that a computer is really good at, such as providing multimedia, storing records, analysis of orders & invoices, tracking systems across time, allowing levels of difficulty. You could mention things such as your solution can be developed in a modular manner – i.e. customers, orders, products or sprites, each object (in the game), levels, introduction, help screens etc.

You can also mention things like repeated code – such as several modules being similar (add customer, add employee or baddie number 1, baddie number 2 sharing similar properties). You could mention the use of libraries and that you won’t have to code everything yourself from scratch.

**Example of the solution description**

*“The shoot 'em up is a subgenre of the shooter genre of video games. In a shoot 'em up, the player character engages in a lone assault, often in a spacecraft or aircraft, shooting large numbers of enemies while dodging their attacks. The genre's origins can be traced back to Spacewar!!, one of the earliest computer games, developed in 1962 and eventually released in amusement arcades in the early 1970s. However, Tomohiro Nishikado, creator of Space Invaders, is generally credited with inventing the genre.*

*These games are usually viewed from a top-down or side-view perspective, and players must use ranged weapons to take action at a distance. The player's avatar is typically a vehicle under constant attack. Thus, the player's goal is to shoot as quickly as possible anything that moves or threatens him. In some games, the player's character can withstand some damage; in others, a single hit will result in his destruction. The main skills required in shoot 'em ups are fast reactions and memorising enemy attack patterns. Some games feature overwhelming numbers of enemy projectiles, and the player has to memorise their patterns to survive. These games belong to one of the fastest-paced video game genres. Large numbers of enemy characters are typically featured. These enemies may behave in a certain way dependent on their type, or attack in formations that the player can learn to predict. The basic gameplay tends to be straightforward and many games offset this with boss battles and a variety of weapons. Shoot 'em ups rarely have realistic physics. Characters can instantly change direction with no inertia, and projectiles move in a straight line at constant speeds. The player's character may collect "power-ups" which may afford the character greater protection, an "extra life", or upgraded weaponry. Different weapons are often suited to different enemies, but these games seldom keep track of ammunition. As such, players tend to fire indiscriminately, and their weapons only damage legitimate targets.*

*Typical games in this genre include: Space invaders, Project starfighter, Galaxian, Defender and Resogun.”*

**Problem Definition**

Cricket is a very unpredictable game and there are many factors that play into a team’s success/loss. Predicting cricket matches is important as it allows analysis to be done on the strengths and weaknesses of a team, and allows the users to be able to vary factors that may change the chance of success of a team. As of right now, predicting the outcome of these matches is a very long and painstaking process, as it is done by hand using Cricinfo, Wisden etc. This means that it takes a very long time, and is prone to numerical mistakes. Computer software that does this is normally paid, or does not include many additional features, just a chance of winning that is normally always skewed to 50% and inaccurate.. Hence, I will be creating an application that the end user can download to predict the outcome, along with other features that other software may not have, such as innings simulation and the ability to customise a lot of parameters, such as the players in each time, the venue etc.

My solution will mainly use machine learning and data analytics to predict the answer, and will make use of web scraping to find news articles and opinions from trusted sources that will provide a reliable estimate. I will use a regression model, to look at past results to provide a numerical “strength of the team”, where more recent results are weighted more heavily than past results, while also incorporating a randomiser to account for random events. This problem is suited to a computational approach as the software must be built on a computer, and a large amount of data must be processed and analysed which cannot be done by hand.

**Why can a computational approach be used?**

The main reason a computational approach is used underpins the problem itself. Since analysis by hand using Wisden and other articles take too long, I will develop this software so that the computer can do all of the calculations very quickly and save time for the user. This is only possible due to the speed of modern processors at processing array data. This also allows for simultaneous processing to happen, so more than one calculation can be done at once through the use of pipelining, which is not possible with human analysis. Human analysis is also prone to a lot of mistakes, which can be completely prevented by using a computer, provided that the inputs are correct. The large amount of data that needs to be analysed further suggests that a computational approach is required. Computers excel at array processing, which allows this to be done very quickly, as well as sorting and searching for data. This large amount of data also needs to be stored. This can be stored either on the cloud or locally, both of which take up a very small amount of physical space compared to paper copies, and allows for the use of databases to improve efficiency.

Storage is another key part of the solution that reinforces the strength of a computational solution. The use of database handling would allow me to provide an interface for the user, allowing them to change various factors manually such as the weather.

As the problem lends itself to a computational approach, there are a variety of computational thinking techniques that can make the problem easier to solve.

**Computational Thinking Strategies**

**Decomposition**

I can split my program into different modules, and use the divide and conquer method to work on these independently. The GUI, regression and analysis can be done in different modules which would make the development process more streamlined.

**Divide and Conquer**

The solution can also be developed in independent modules, such as the machine learning model, GUI, database storage all being modules that can be worked on independently and then incorporated into the main program to maintain readability and structure. This is suited to a computational approach as different files and programs can be built and coded separately, all of which can be imported as modules/libraries into the main program. The use of modules and libraries also allows for lots of code to be reused, which can save time and processing power.

**Abstraction**

In a real cricket match, there are a lot of factors that play a marginally small role in the outcome of a match. This can include the wellbeing of players, time etc. These factors would be practically impossible to emulate in my program, and hence I will only include factors that have a chance to play a vital role in the outcome. Furthermore, the machine learning algorithm and the analysis can be hidden from the user so there is no need to develop a GUI for these parts, only for showing the results.

## (ii) Stakeholders

*(a) Identify and describe those who will have an interest in the solution explaining how the solution is appropriate to their needs (this may be named individuals, groups or persona that describes the target end user).*

Who is it for? What organisation/genre of person. Give some detail e.g. “I am designing a game to appeal to 14-16 year olds who are interested in sci-fi” or “My organization is a restaurant in the Cotswold village of Moreton-in-Marsh, they serve 14 customer per day and have 3 main meals…”.

It’s far easier if you can name a particular person who will providing feedback as you develop your solution. This means that they can comment on your work – ALL the way through your project. So pick someone that you can nag!

In this section, if you have an organisation you need to talk about what they currently do, why it isn’t working and how they intend to use your system in the future. How will it help them? Will it save time, combat existing inefficiencies? Allow them to track data? Stop redundant data? Allow multiple access to data? Allow backups etc. You must be detailed, don’t just say that it will improve things, tell me what things. You could also include things such as cooperate logos, colours etc

If you have a game for a particular audience, make sure that you have thought about what you can and can’t include in your game. If your audience is young, you will have to think about colour schemes, language, playability the kind of things that they can and cannot do. If your audience is older you also need to include things like why they would like the game, have they always wanted a game that….? Are they the greatest fan of …..? You may want to cross reference your work with the PEGI rating system. You may want to mention that you will have to design the levels to increase in difficulty or even have a tutorial level that presents instructions to users (pants for under 8s, but useful for over 16s).

You could also talk about the kind of platform that you are writing for. Does your organization have three laptops? Does it have two desktops? Are you designing for a phone, tablet? Are there resolutions that you can or can’t use?

**Stakeholders**

The main stakeholders of this program would be my end users with access to a computer. Since there is no potentially harmful content in my project, there will be no age limit to be able to access the program. The user interface would be very simple, and all of the features will be very easy to use, so experience with a computer is not required. The hardware requirements for my stakeholder is that they would have to have to be able to access a modern computer that is connected to the internet, with input and output peripherals, such as a monitor, mouse and keyboard.

I have selected a group of 4 stakeholders that I can easily contact, that regularly follow cricket. 3 of them are in the 16-18 demographic and follow cricket to varying levels. They are Alex Varghese, Yachna Dhir and Guneet Gill, who have mostly used some sort of prediction or fantasy app before (fantasy cricket world cup, cricbuzz etc). Guneet also plays cricket regularly, and I can ask him about how various factors play into the outcome, venue, toss, dew etc as he has experience playing. The last stakeholder is my father, who is an avid supporter of the Indian cricket team. He is also a software engineer, so I can regularly ask him about the features that would make a program lightweight and easy to use.

I will be sending stakeholders forms regularly, that they can use to provide any input into the development process, while also providing initial conditions at the beginning of the development process.

**Stakeholder Survey**

I sent a survey to each of my stakeholders asking them 13 questions that will help me define the functionality and requirements for the program.

**Question 1 - How often do you use fantasy apps?**

Everyone responded weekly for this question, showing that there is high demand for these kind of apps, and that my project would be used regularly by my stakeholders

**Question 2 - How often do you follow cricket?**

The majority responded daily, while 1 person responded weekly and 1 yearly. From this, I am able to diversify my stakeholders, and make it such that it will be interesting to people who don't follow cricket on a daily basis. And hence, I will have to minimise the use of jargon in my program, so that every end user can understand it, as some may not be that knowledgeable about cricket

**Question 3 - What is your age?**

The majority were 16-18, suggesting that I need to tailor my program to meet the requirements and characteristics of the adult/young adult demographic.

**Question 4 - Would you be interested in a feature that predicts the “impact player” for a given match?**

Everyone answered yes to this question, suggesting that it is a highly demanded feature, and I will put it into a high priority for functions of my program.

**Question 5 - What platform would you prefer?**

3 people responded that they would prefer a website, as no installation and compatibility is required, and they can immediately navigate to the website from their browser instead of having to open a new application. Hence, I will create my program on a website.

**Question 6 - Would you be interested in being able to simulate every ball in a match?**

3 responded no, as they would find it too time consuming. To make my program lightweight and streamlined, I decided to not include this feature.

**Question 7 - Would you want a scorecard displayed at the end of a match?**

The majority responded yes to this question, and hence I will create a feature that will output the scorecard as a downloadable txt to the user so they can view who scored the runs, who took the wickets etc.

**Question 8 - Have you used a cricket predictor before?**

Most people said no to this question, or simply said google win prediction. This indicated that there was no exposure for accurate win predictors, unlike the google one. Because of this, I will also have to make my program very easy to understand and use as they have never used a predicting software before.

**Question 9 - Would you be interested in a feature where you can drag and drop players from different teams?**

Everyone answered yes to this question, suggesting that is a highly demanded feature. This will introduce more functionality into the program, and can mean that end users will spend more time with the program experimenting with different teams.

**Question 10 - Would you want to manually vary pitch conditions?**

A majority of people said that they wanted to vary the toss and venue, as they play the largest role of the external factors when deciding on the outcome of the match. 2 people also said they would want to vary the temperature and pitch condition, but it is likely that this could fall under the various characteristics of each venue. Only 1 person wanted to be able to vary the dew factor in the program, so I will not include that in the program

**Question 11 - Are you paying for any of these prediction software**

Everyone responded no, and hence I will base my research off of free software to make sure that nothing is locked behind paywalls and such when finding good and bad parts of similar software in my research.

**Question 12 - What colour scheme do you want for the software?**

A lot of responses of blue and white resulted, with a large number also preferring a black colour scheme. To make sure that every stakeholder has a colour scheme they would like in the program, I have decided to make my colour scheme a combination of blue, white and black

**Question 13 - Are there any other features that you would like?**

3 responded with new features they would like to see in the program. I had one response that wanted the resultant scorecard to be emailed to them after they have simulated a match, which could be useful if they want to look at it on a different system. 2 people responded that they wanted my program to be able to predict the best possible fantasy team from all of the player pool. This would add a lot more functionality and could help them with choosing a fantasy team in the actual fantasy competitions. Lastly, 1 person wanted the program to be a fantasy app that includes prediction, however, I felt that this is straying away from the problem, and would be too difficult to maintain. Therefore, I decided to not include a fantasy game inside my program.

With my survey, it allowed me to specify initial requirements that would help me in starting the development process on my first prototype.

**Stakeholder access**

To prevent my code from being accessed and biased, I will include some rules that I will make the program to follow.

* None of the end users will be able to make changes to the dataset, as the main aspects will be hidden in the source code, and hence only the results will be shown to the user. This is to stop the results from being skewed and biased on purpose by the end users.
* The database from which the data is retrieved from will be stored centrally as a result.

## (iii) RESEARCH THE PROBLEM

*(a) Research the problem and solutions to similar problems to identify and justify suitable approaches to a solution.*

*(b) Describe the essential features of a computational solution explaining these choices.*

*(c) Explain the limitations of the proposed solution.*

When you conduct your research you will have to add detail. For creating a game, this is relatively simple. Think about colours, levels, playability, what did you like, what did you hate, what can you include, what is just beyond your capability, what extensions could you add. You must describe the games that you test and then evaluate each one – I would suggest that three would be sufficient. Go into detail - What are the controls? Who are the enemies? What is the aim? What is the jeopardy? How do you progress? Are there lives? Is there a score? What are the graphics? What platforms is it available for? Are there sounds? DO you have weapons & how are they used?

If you are looking at industrial software, this can be harder. If you can get hold of demo software, then use this and see how it works. You may have to think slightly laterally. For example, if you are developing a booking system for theatre tickets, you may have to look at similar industries, such as cinemas, events & travel. You may have to rely in the blurb that they have on the website – with video adverts etc. Include screenshots so that your examiner gets an idea of the software that you have been researching. But your analysis will be similar to those looking at games. What features does it have, which can you do, which can’t you do, which do you like and those that you don’t.

You must also mention what you cannot do – creating a limitations heading would be good. What will you be unable to do & why. E.g. I will develop my system for a Windows environment, because I only have access to Windows based computers. I will have to limit my levels to 7, because I won’t have time for 300. I will have to encrypt my data using a simple Ceaser Cypher, because it will take too long to write an encryption algorithm.

You must also include some information about the computational solution. Here is an example:

### *Abstraction and visualisation*

* *What are the key objects in your game?*
* *What sprites, symbols, icons or font sets will be needed?*
* *Are there further lines of investigation to follow?*

### *Thinking ahead*

* *•What will your inputs be? e.g. cursor keys, WSAD, mouse etc.*
* *•What will your output screens be? e.g. main menu, story, controls, level select, game screen, game over screen, high score.*

### *Thinking procedurally*

* *•Identify the 'game states' you will need. e.g. menu, load level, playing, game over.*
* *•Break the problem down into what will happen in each of the game states for OnCreate(), OnUpdate(), and OnRender().*
* *•Pipelining: Decide the order of each component of the game. e.g. in 'playing' game-state: get inputs, move object, collision detection, movement correction. In OnRender() draw player, draw missiles, draw enemies.*
* *•Explain that the program loops continually around OnUpdate and OnRender.*
* *•State that these sub-problems make the project amenable to a programmed solution.*
* *•Will you have any reusable program components? Functions used for more than one purpose.*

### *Thinking logically*

* *•What are the critical if statements?*
* *•What are the critical loops?*

### *Thinking concurrently*

* *•Are there aspects of the game where more than one thing happens at once? The classic example is the sound player using multi-channel sound to play different sound effects and background music at the same time as updating the game logic.*

**Similar software**

**Google win predictor**

The google win predictor is not really a piece of software, but is built into the website when you search for a cricket game that is about to happen in the future

A screen shot of a video game

Description automatically generated

As you can see it just provides a win probability, and does not say anything about how it calculated the win probability. But, it does seem fairly accurate as India is playing at home so they would have an advantage against New Zealand. Further proof that it is fairly accurate is the win predictor for Australia vs Netherlands. A screen shot of a game

Description automatically generated

Australia is a much stronger team than the Netherlands, and hence the win probability is justified. However, the fact that they do not explicitly say how they calculated the win probability is a bit concerning, as it may just be updated by a person instead of using a computational method.

However, a good point from this win predictor is that it updates live as the match goes on. This means that it is dynamic and can respond to changes in the match by looking at the required run rate, overs left, the number of wickets left and the original starting probability. This dynamic feature would be nice to implement in my code along with the scorecard, with it updating every over.

The cricket win predictor is part of Google's larger sports analytics ecosystem, which includes search integrations, Assistant functionalities, and dedicated sections on YouTube for highlights and analyses. This integration ensures that insights from the predictor are widely accessible across Google's platforms. As with any data-driven tool, there are potential concerns regarding privacy and the use of personal data. Users may be wary of how their data is collected, stored, and used within Google's ecosystem.

**Advantages**

* Probability bar changes colour depending on the team (yellow for Australia etc)
* A visual probability bar is displayed rather than just numbers
* Very lightweight and easy to use
* Dynamic, so it responds to changes in the score
* Can easily be accessed on a web browser

**Disadvantages**

* Too simplistic
* Lack of functionality
* Does not provide a reason from where the estimate comes from
* Only limited to matches in the future that will actually take place
* Only occurs for large scale matches

**Cricviz**

Cricviz is an industrial software for data analytics for cricket matches. It is used by large cricket organisations, such as Cricket South Africa and the Hundred. Since it is being used by so many big organisations, it is clear that it has lots of functionality and is very accurate.

A close-up of numbers

Description automatically generated

It also has a large dataset, as it has collected lots of data from different matches that have been played around the world, so the uncertainty from the results from CricViz will be very low.

Cricviz provides a wide range of services that are all related to prediction software. One of its products is their flagship - **Centurion** that provides deep analysis on upcoming matches.

Centurion is their platform where pre populated panels of models, and data analytics are shown, such that they have access to player statistics and more.

It includes match centre, video centre, analytics and player statistics sections

A screenshot of a computer

Description automatically generatedA screenshot of a video game

Description automatically generated

I really like this product as it has large amounts of functionality and it contains all the features an analyst would want to have when analysing a specific team or calculating a win probability. It has a vast amount of visualisations, as well as visual records of each ball so that it is very easy for the analysts to find patterns in the data, and make their calculations based off of those patterns. The match centre provides a dynamic updating win probability, as well as player impact and a multitude of other features. The player profiles section also highlights the strengths and weaknesses of each player so coaches and management can easily identify what to work on.

However, in terms of my project, I am only looking to create a win predictor, and predict the best possible fantasy team. On CricViz, there are too many features that could result in stakeholders getting overwhelmed if they just want to predict matches leisurely. Whilst the general content is excellent, there is also quite a lot of jargon used in CricViz, which may deter stakeholders from being able to understand and use the program, as CricViz is intended for industry analysts, not leisurely cricket fans. A query tool is included in Centurion as well, which provides archives and player statistics. Once again, this will not be relevant to my project. Finally, they have a team selection software that aims to find the optimal squad for a team to play in a certain venue/conditions against a certain opponent. This is very similar to predicting the best fantasy team a team can play, only just a bit more complicated.

**CricViz Match Centre**

The win prediction features which are most closely related to my project are inside the match centre.

A screenshot of a computer

Description automatically generated

The WinViz algorithm is similar to my machine learning algorithm which can display the current chance of winning for a certain team by looking at past results, as well as predicting a resultant scorecard. However, the PitchViz feature is very impressive as it quantifies the pitch conditions on a scale that makes it easy for the analysts to read and visualise. However, this will be very difficult to pull off as ball tracking data is required, which is not publicly available. The main issue with CricViz for my stakeholders is the price, as CricViz is a paid software.

To make its complex data more accessible, CricViz offers sophisticated visualisations and dashboards. These tools allow users to interact with the data, making it easier to digest and apply, whether for broadcast analysis, team strategy meetings, or fan engagement.

**CricViz advantages**

* Includes a vast amount of features to help analysts
* Is dynamic (win % changes as match progresses)
* Includes video ball tracking for pitch conditions analysis
* Large amounts of data visualisation
* Very accurate (used by cricket boards)
* Split into modules, and can easily navigate product
* Query tool and database included

**Disadvantages**

* Includes a vast amount of jargon, so stakeholders may not be able to understand
* Is locked behind a paywall
* Contains too many features for an average cricket fan, so they may be overwhelmed
* Could not find an option to drag and drop players from different teams to simulate their own matches
* Requires an install and specialist help

**Fantasy Premier League**

The fantasy premier league (FPL) is a software that allows players to act like their own manager and buy different players with a set budget that they can put inside the squad. Points are scored based on how the players do in the real world, and their match rating is considered. While this is not exactly a cricket prediction software, it has some elements of prediction as well involved with the player picking, as different players have different prices depending on how well they have performed recently. Players can be transferred every week, and different managerial choices can be made which can boost points, such as making the best player captain and so forth.

A screenshot of a football game

Description automatically generated

Here I have included a snapshot of the FPL website. The website is very sleek and there is no unnecessary detail apart from the sponsorship logos. The pitch view is very nice to look at, with shirts being used as icons for each player and the position being displayed underneath. However, this is much harder to accomplish on a cricket predictor since there is no “formation” for the team and I would have to use something like the list view option on the FPL website.

A screenshot of a football game

Description automatically generated

The option to add players is very well written as it includes filters for the program to search for different players. In my program, I could do something similar by looking at the ICC ranking of each player. However, there is no option to filter by nationality which is something I would like to include in my program. This menu is very visual, but I don’t think I could replicate the shirts and different jerseys on my actual program given the time constraints I am under. The budget option is unnecessary with my program so I will not be adding it.

A screenshot of a computer

Description automatically generated

Here you can see the login page for the FPL website. It consists of a sign in label, with the input boxes to the right of the label. It consists of a sign in submit button, with an option to reset your password if you forget it. I will not be replicating this feature in my program, as this would require signing in with an email address/phone number such that the account can be recovered. This would require setting up networking/mail servers such that an option to reset the password is shown. I do not wish to implement the login with Google/x etc options as they would be too complicated to build in the given scenario. This would then require its own interface, and would require networking to access the SQL database. This would take too much time and would not increase the usability by a large factor. Hence, I will not be including this feature in my program due to its complexity and the time constraints I have on the program However, I do not like excessive advertising to use the product, as can be seen in the text to the left of the sign up button. Hence, on my login page, I will be using minimal text to make the interface as simple as possible.

A red and white rectangular object with white text

Description automatically generated

This occurred after I typed in an incorrect username/password combination. I like this colour scheme as it is very visual and the alert box is very easy to read - I will aim to replicate something similar in my program as well.

A screenshot of a computer

Description automatically generated

This is input validation that seems to be done on the client side, as it displays a different error message to the previous one. This is a format check, and can be done on the client side to reduce the load on the server, and decrease waiting times for the server to provide a response. I can include a similar check in my program that will perform a presence check on both fields before sending a request to the central database.

A screenshot of a computer

Description automatically generated

This is the sign up page for creating a new account for the system. Since FPL is on a much larger scale than my program, it includes lots of information such as first name, last name and gender that can be excluded for my program. My program will only look at the username and the password, email address and confirm password.

A screenshot of a computer

Description automatically generated

When empty, an error message is displayed underneath the input box. The use of the cross icon makes this highly visual and I want to include something similar in my program as well.

A screenshot of a computer

Description automatically generated

The password must include checklist is very well done, as it shows which conditions are failing on the password, such that it can be distinguished if more than 1 condition fails, unlike a general password invalid error message.

Overall, I think the login/signup pages are very well done, and I will be taking a lot of features from these and will include them in my final program. However, I will scale this down for simplicity, and use a different aesthetic that will conform to the rest of my program.

By doing some research, I have also read about what type of variables and factors the FPL algorithm uses .

Metric calculations

FPL analytics use a wide range of performance metrics for players, such as goals scored, assists, clean sheets, and even subjective measures like influence, creativity, and threat (ICT) index. These metrics can be adapted to cricket, focusing on runs scored, wickets taken, catches, strike rates, and economy rates, among others.

FPL enthusiasts often use predictive models to choose players expected to perform well in upcoming matches, considering factors like form, fixtures difficulty, and historical performance against certain teams. For cricket, similar models can predict player performances based on opposition, venue, weather conditions, and recent form.

Managing a team within a budget and deciding when to transfer players in and out is a crucial part of FPL strategy. This aspect can translate into a cricket predictor program by considering player valuations and the impact of team changes on predicted outcomes - however, this is not a focus on my game as my stakeholders are not team managers/fully serious about the program and would just like to see some results and have fun. Hence, I will not be adapting my program to make it easier to make transfers/any managerial activity.

A screenshot of a sports schedule

Description automatically generated

The FPL website also provides prior statistics on each player and a photo. Since I am not planning to update my program with each match in real time, I will not be including this feature and keep it more simplistic as adding this detail would not really help with the main purpose of the program.

**FPL Advantages**

* Very visual and easy to use
* Filters to drag and drop different players
* Free
* Updates in real time and is interactive
* Very concise (functionality covered in only 1 page)
* Login and signup page very visual and easy to interpret

**FPL Disadvantages**

* Have to sign in to use product
* Advertisements and sponsorship logos seem clunky
* Too much detail and statistics available to the user
* No option to simulate your own matches
* Player selection restricted under a budget
* Cannot sort players by nationality

From these three software, there are a wide variety of features that I can include in my program

This will include:

* Simplified language to avoid jargon.
* Split into modules.
* Right number of extra features (not too simplistic, but not too complex)
* Theme changes depending on teams being simulated.
* Filters for adding new players to the team
* Use cricket icons to make the software more aesthetic and visual
* Restrict simulation to 1 format for ease of use and time constraints

**The proposed solution**

My solution will consist of several parts to the program, which are developed independently to each other. The main features will be:

* Win prediction algorithm.
* Drag and drop different players to form an XI (players are filtered)
* Prediction of strongest possible XI before simulation on a new page
* Venue as input before simulation
* After simulation, scorecard available to download and also emailed to end user
* Option to quit program or restart a different simulation after another has ended
* Welcome page after login/signup explaining how to use the program

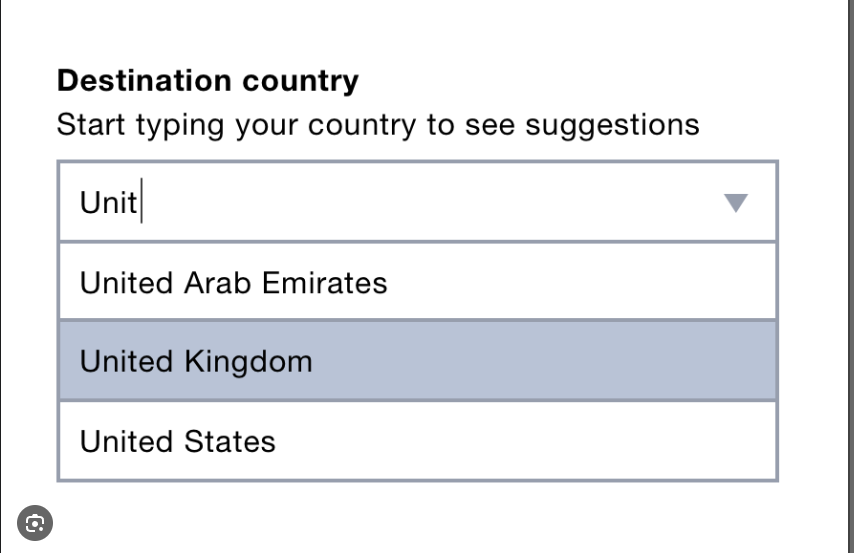
Since I am using a computational approach, the solution is amenable to computational thinking strategies to streamline the development process.

For my project, I will consistently use Segoe UI as it is very clean and professional. In terms of images, I will require images such as stock player images, wickets, cricket balls, bats and more to make my program very visual and easy to use.

**Thinking ahead**

The program will use a quantitative approach to machine learning and will include factors such as date, time played, temperature, and will have a look at layer statistics. It will assess each player's record against the various factors and will weight each factor as required. The model will then finally predict by adding up each players score to finally build a team score as well. The bowling side will also be given a strike rate, average with match ups against each batsman modelled by bowler-batsman past records.

I will have multiple stages to my program, and it will be developed in modules. The inputs to my program will be done with a keyboard and mouse using text boxes that will provide suggestions for the desired player. Like this:



This is how the players and the venue will be inputted.

The output screens for my program will include an original start up screen, where end users can choose between finding the strongest fantasy team or completing their own simulation. If they want to find the strongest fantasy team, the output screen will just be a list of the best players to pick, and an option will be given to quit the application or to restart. If they want to create their own simulation, the input screen will be shown with all the inputs. After the simulation has finished, the results screen will be shown, with an option to download/email the scorecard on another screen. The other option will be to quit or restart.

Hence, I will have to take the email address as an input in order to email the scorecard to the user. I will also be storing the most recent scorecard by a certain user in a file that can be accessed so that they can access their most recent simulation.

Another factor that I will have to look into is the form of the logins database. If passwords are stored as plaintext, another user can sign in to a user’s account if they simply view the database. This is not very secure, so I will be implementing an encryption algorithm such that the passwords cannot be understood.

**Thinking procedurally**

My program will follow a structural cycle

1. Login page
2. Welcome page that shows how to use program and menu to navigate to different pages.
3. Input page (for both simulation and predict best team)
4. Machine learning algorithm completes
5. Results shown
6. Results/scorecard available to download/email and view on a page
7. Ending screen, which can either stop the application or loop back to stage 1/2 as desired if user decides to logout or restart prpgram

This means that the program will be developed in modules, and I can focus on each sub problem by decomposing the initial problem. Since I will be using similar functions for both my “strongest fantasy team” and “simulation” paths, I can reuse a lot of functions and algorithms using the computer, which allows for a computational solution to be built.

**Thinking logically**

I will have a lot of inputs that will include if statements, that can lead to different parts of the program. My start page input will lead to one of two branches depending on the input. For each of those, the inputs from the input pages will be used in if conditions inside the machine learning algorithm so it can contextualise the data provided by the inputs. An if condition will also be used to determine if they want the scorecard to be emailed, downloaded, both or neither after the simulation has finished. The last major if condition will be the end page, where the user can choose to restart the program or end the application. If the user decides to restart, there will be a loop that will go through the program such that it begins from the start again.

**Thinking concurrently**

Since the machine learning algorithm may take some time to display the results, there is an opportunity to use concurrency in my program. Whilst the machine learning algorithm is running, I can start to display the GUI for my program already and load all the extra libraries and modules that will be used for the analysis later. This will mean that the user must wait for less time and will lead to a more seamless user experience.

**Limitations**

Some of the features from the programs that I have looked at will be impossible to develop and replicate effectively in the given time. For example, a possible fantasy game like the Fantasy Premier League for football has too many features, and the code and databases need to be updated regularly, and new data needs to be received after a match has been played. This would be too difficult to accomplish in the period that we have for this period, and hence I will not be including it in my program. Another limitation to my program is the use of rolling data in my simulations. This would mean that the data would have to be regularly updated after each new match, which would involve a lot of API calls and networking. I think this would be too difficult to learn and apply before the deadline, and hence I will only use 1 set of the most recent data, and possibly update the dataset and retrain the model after a certain time, maybe around every 3-6 months. Since I am using WinForms for my project, resizing will be an issue and hence I will be making the size of the form a fixed size, but that size will also mean that it should be visible on all display screens. Hence, it may appear to be different sizes on a QHD monitor compared to a FHD monitor, but all features should be able to be accessed regardless. This will mean that I must restrict the form so that it cannot be resized.

## (iv) SPECIFIY THE PROPOSED SOLUTION

*(a) Specify and justify the solution requirements including hardware and software configuration (if appropriate).*

*(b) Identify and justify measurable success criteria for the proposed solution.*

Success Criteria

Aesthetic Objectives:

* a. The window will be 1100px by 600px and will not be resizable.
* The background colour will be blue for all screens.
* b. A blue, black and white colour scheme will be used in accordance with stakeholder requirements.
* c. The font used throughout will be Segoe UI, and the size will be consistent with the size of the text boxes. Occasionally, another similar font may be used. Player names will be small while titles will be large comparatively.
* d. The background colour for text boxes is white, and the text inside buttons will be black
* e. Input boxes will be white, with placeholder and inputted text being black
* f. The title of the page will be at the top for each page, and there will be a footer at the bottom for each page.
* g. Error labels should be to the right of the relevant input box, and they should be the same size font as the labels/input boxes, but should be red in colour, and invisible if there is no error.
* g. Welcome page
  + The menu page will have 2 buttons near the top to navigate to the “predict best fantasy team” (PBFT) or the simulation page
  + There will be a button near the top right-hand corner of the page to exit the application, which will be the same for all pages.
  + There will be an option to download the most recent scorecard simulated by the account, that can be accessed by clicking a button towards the right of the screen.
  + The title will be blue and will display “Menu”.
  + There will be a logout near the top left corner of the screen that will return
* h. Signup page
  + There will be 4 input boxes with the order being email address text, username, password and confirm password to ensure that the user does not misinput into the password field.
  + The input boxes will be vertical arranged and aligned. For each input box, there will be a label to the left to indicate it’s a username, password etc.
  + For each input box there will also be an invisible error label to the right, that will display the relevant error message in red.
  + The username input will contain placeholder text that displays sriv123 and this will also be true for the email address, with placeholder text being [sri@gmail.com](mailto:sri@gmail.com) (not real email).
  + There will be button underneath the input boxes which is the sign up button. When clicked, if the inputs are valid (username/email not already in database etc), it will create a record in the database with a username and a password. Then, the user will be on the main menu with the account they just created. When it is clicked all error messages should be cleared
  + If the inputs are incorrect, a relevant message will be displayed in the error labels to indicate which error has occurred and in which field the errors have occurred.
  + There will also be a button next to the register button, that will act as a clear button. This will clear the values of all fields.
  + There should be a show password checkbox underneath the confirm password input box that will show the password when checked and hide when unchecked. The password is set to hide as default. This should automatically be unchecked when the clear button is clicked.
  + This page should be accessible if the user decides to logout.
* i. Login page
  + There will be 2 input boxes with the order being username and password.
  + The input boxes will be vertically arranged and aligned. For each input box, there will be a label to the left to indicate it’s a username/password
  + For each input box, there will be an invisible error label to the right, that will display the relevant error message
  + The username input will display placeholder text of sriv123
  + There will be a button underneath the inputs, which will give an option to register, and this will be highlighted when hovered over. When clicked, if inputs are valid, it will login into the main menu on the user’s account.
  + There will be button underneath the input boxes which is the login button. When clicked, if the inputs are valid (username/email not already in database etc), it will check against the records in the database to match the user to an account. Then, the user will be on the main menu with the account they just logged in with. When it is clicked all error messages should be cleared
  + If the inputs are incorrect, a relevant error message should be displayed in the error labels to indicate which error has occurred and in which field the errors have occurred.
  + There will also be a button next to the register button, that will act as a clear button. This will clear the values of all fields.
  + There should be a show password checkbox underneath the password input box that will show the password when checked and hide when unchecked. The password is set to hide as default. This should automatically be unchecked when the clear button is clicked.
  + This page should be accessible if the user decides to logout.
* i. Simulation page
  + The title will be blue and will display “Simulation”, and the page will be split into half down the middle of the page.
  + It will contain 11 text boxes to input players.
  + Player input boxes will have an icon with the respective position e.g. a wicketkeeper may have a pair of gloves next to it.
  + Player input boxes will contain a filter for nationality, ICC ranking that will use similar buttons/input boxes to control
  + Each input box will select a player and their name will pop up in the input box after they have been selected. With the input box, there will be an option to also remove the current player.
  + Duplicate players must not be added, so that no player can be repeated in the 22 players.
  + There will be a button at the bottom of the input boxes that will confirm a submission.
  + When both submissions are confirmed, the simulation will start, and the scores of both teams will be outputted, and the individual scorecard will be omitted.
  + Another button below the scores will give an option to reset the current simulation and remove all selected players from their slots.
  + The scorecard can be accessed from the main menu through the button as previously mentioned.
  + Once again, buttons to the menu page and PBFT will be at the top of the page near the title.
  + There will be a final smaller menu inbuilt into the simulation page where users can alter the pitch conditions, toss etc.
* j. PBFT page
  + The title will be blue and display “Best fantasy team” and it will be at the top of the page.
  + There will be a button in the centre of the page that will initialise the prediction.
  + There will also be 11 empty player slots on the screen, with icons to show the role of each player.
  + Once the algorithm is finished, the names of the players will be assigned next to their respective icons.
  + Lastly, there will be a button towards the bottom that will restart the algorithm.

Inputs:

* a. Screens are navigated through buttons on the screens, and a navigation menu near the top, and some functionality is also controlled by these buttons as well.
* b. Text entry fields:
  + Simulation – Add each player to a slot by typing in their name and will provide a drop-down menu as discussed earlier. This input is a name that will be validated and cleansed to ensure that no erroneous data is inputted.
* c. Non navigation buttons
  + Used to control confirmation of a submission/scorecard/start simulation as well as swap the current player/remove/restart simulation in the Simulation page.
  + PBFT page used to reset/start prediction.
* d. Username and password will be inputted into the login page, checked against the database and will enter the program/fail depending on the inputs
* e. Username and password will also be inputted into the sign-up page. The username and email will be checked against the database to make sure it is not already in the database or empty, and the password will have a minimum of 8 characters and 1 number and 1 special character. The password characters should appear as \*\*\* when typing in password and confirm password. Password and confirm password should match for signup to be successful.
* f. In the signup page, email address will also be inputted. A format check is required such that it is in the form [sri@gmail.com](mailto:sri@gmail.com), with the @ and the domain. This will be used to mail the scorecard to the email address, and hence will also require for the email to be existent.

Processing:

* a. There are two main functions being used in the program – the PBFT function and the simulation algorithm. Either of these will need to be used depending on the page they are in.
* b. Simulation
  + The simulation algorithm will be run, using the 22 selected players and the conditions as the arguments and will return the scores and the resultant scorecard.
  + The input validation will be done outside of the function.
* c. PBFT
  + The simulation algorithm will run, and the only parameters will be the pitch conditions, venue etc – 11 players with different roles will be returned

Hardware/Software requirements:

* Must have a 4-core processor and above that can run Windows
* Must be connected to the internet.
* Must have a suitable web browser – Chrome, Safari etc.
* Must have a mouse/touchpad/keyboard for inputs.
* Must be able to run Python, SQL, .NET and sklearn, TensorFlow, pytorch, pandas must be pre-installed into the Python interpreter.
* Must have more than or equal to 4GB of RAM.
* Must be able to run SQL/SQLite

You must have detail here, because this will form a substantial part of your evaluation and is also your success criteria. If you just say “I will create a game”, then your evaluation will be “I created a game”!. You will also need to list the hardware and software requirements. This will include stations (laptops/PC.Tablets etc), the OS, any software needed (e.g. Python), any peripherals (printers etc) and networking.

**Here is an example of solution requirements:**

Following this feedback, I have made some changes to the design objectives. The new objectives can be seen below. The altered parts have been highlighted.

1. Aesthetic objectives
   1. The window size is 800x600 pixels and the window should be non-resizable
   2. The colour scheme is a light blue background with white and yellow text
   3. The monster character will be a dark blue/purple colour with a hair-like texture
   4. Text boxes where text must be entered will be filled white rectangles
   5. Text in text boxes will be black
   6. The font used will be Arial Rounded MT Bold or similar and will be consistent throughout the software
   7. Action buttons
      1. These have yellow text in all capitals
      2. They have square corners
      3. They will have a slight raised effect
   8. Quick Fire Questions
      1. The questions appear one at a time
      2. The questions are white and in large writing
      3. The answer is entered in a text box at the bottom of the screen
      4. Right answer – ‘Correct!’ is displayed on the screen in green writing
      5. Wrong answer – ‘Incorrect!’ is displayed on the screen in red writing
2. Inputs
   1. Screens are navigated and actions are carried out by clicking on-screen buttons
   2. There are text entry fields on the following screens:
      1. Student Log In
         1. Information to be entered - username
      2. Teacher Log In
         1. Information to be entered - password
      3. Add Student
         1. Information to be entered – first name and last name
      4. Delete Student
         1. Information to be entered – student username
      5. View Students
         1. Information to be entered – student username
      6. Game Screen
         1. Information to be entered – answer
   3. Check boxes are used to select times tables
3. Processing
   1. The software will need to change the questions used in the game depending on the times tables selected
   2. Quick Fire Questions
      1. Random questions will be generated
      2. The software needs to determine whether the given answer is correct
      3. The software needs to record the number of correct answers given
      4. A timer is used to time how long it takes for 20 correct answers to be given
   3. The software will need to update the high scores if necessary at the end of the game
4. Outputs
   1. Quick Fire Questions
      1. The question is displayed
      2. The number of correct answers out of 20 is displayed
      3. The timer is displayed
      4. Whether an answer was right or wrong is displayed
      5. The final score is shown at the end of the game
   2. High Scores
      1. Displays top 5 times for the game
   3. View Students
      1. Displays list of all student usernames on screen
      2. Must be able to display at least 45 students
      3. If a username is entered, a pop up appears with that user’s details
   4. Pop ups are used to alert the user if a wrong username has been entered, or if a field has been left empty
      1. These should have a design consistent to the main program window

# (2) Design OF THE SOLUTION [15 marks]

## (i) DECOMPOSE THE PROBLEM

*(a) Break down the problem into smaller parts suitable for computational solutions justifying any decisions made.*

**Python**

For the actual machine learning algorithm itself, I will use Python as it is the industry standard when it comes to data analytics/deep learning. This means that it is very well suited to the work I will be doing. Another reason for choosing Python is that many tutorials to learn how to create the ML algorithm are done in Python, as it is the industry standard. This will make learning the skills to create the program easier to accomplish. Python also contains lots of libraries for this purpose, so I can reuse lots of code in my program, which will save time and be more efficient for the end user. The libraries used in Python make file handling rather simple and have inbuilt integration with database software such as SQL and SQLite.

**Github**

Another feature I will be using is that I will maintain my project in the cloud using Git. I will create a local repository which can be then pushed onto the cloud and then changes can be staged, committed, pulled/pushed to keep different versions of the project the same. The use of this version control would also allow me to access and change the project from different local machines, which would mean that I could edit my project from various places, such as from school and from home, without having to worry about saving it in a cloud/emailing as it will be controlled and any errors will be managed by the software itself. I am choosing Git as my source control as it is widely used in industry, and there is direct support built into the IDE that I am using – VSCode for Python and Visual Studio for .NET. There are extensions for Git in VSCode and Visual Studio that provide autocomplete features as well as more functionality which I can use to make programming more productive.

**.NET Core with C#**

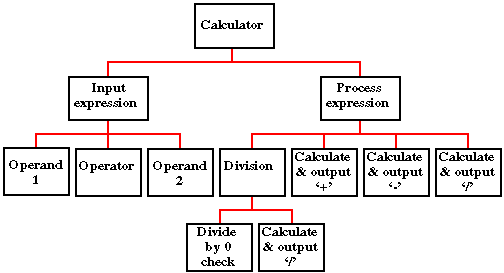
I will also be using the .NET core to build my application and provide a GUI directly on Windows, which will run on C# using WinForms. I am using .NET to build my application as it is very commonly used in industry. It contains features which are directly supported by Windows, making designing buttons/other parts simple. This is because buttons are built as modules, so they can be redesigned very easily and reused very simply compared to coding a button from the start itself. .NET Core is ideal because it keeps user data safe and also has a strong community of developers and lots of guides and help available. This makes building the app easier because you can find solutions quickly and reuse working code that other people have built previously. It is a RAD (rapid application process) so it is very easy and quick to develop applications

**SQL Database and csv files**

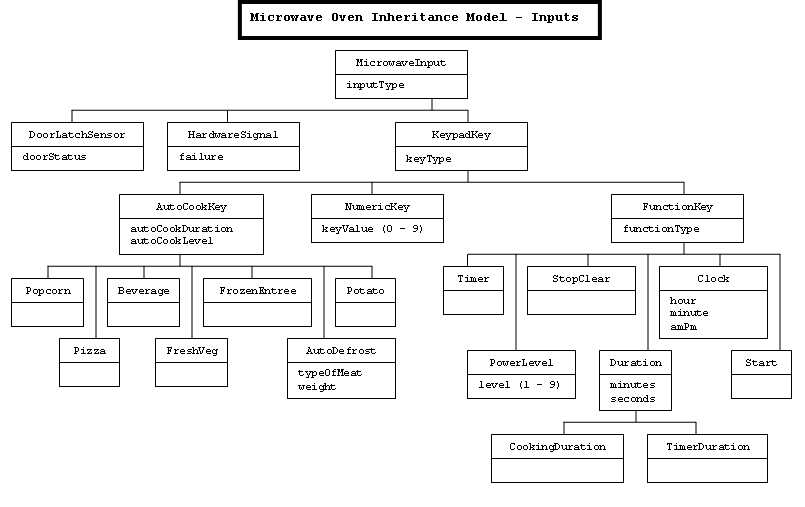
I will be using a SQL database to store temporary data in calculations and the user accounts information. I will be using SQLite to accomplish this as it is very easy to set up and start working with compared to a full SQL server, which adds more complexities to the problem and is not necessary for this scenario. The advantage of using SQL is that it is specifically made as a database management system, and the searching and sorting algorithms are very fast compared to Excel. Complex queries can also easily be carried out in code which also makes them much more suited for this program. SQL servers are also used to maintain ACID – as they apply features such as record locking to ensure that ACID principles are maintained throughout for a consistent database with integrity. I will meanwhile be storing my historical data as csv files as pandas (the python module for machine learning has large amounts of functionality built for csv files, so the data analysis will be very easy to accomplish. It is also the industry standard when it comes to these types of programs.

Choose and justify your choice of language – it is NOT good enough to say I will use Python cause school have it and I can use it. That can be mentioned, but you should say – I will use python because it has libraries that I can utilise, such as….I am not too concerned about the GUI element of my solution…..The file handling in Python is relatively easy…..I can adapt Python to become an OOP and this will help me because – this class will contain this object and this object and this object, reducing the amount of code that I need. I will use python in a procedural manner because my project will achieve this task and then this task and so on which lends itself to the procedural nature of this language.

Include a top-down diagram (or inheritance/class diagrams for OOPS) to describe your project and how it will interact. You could include other DFDs – including Entity diagrams Activity diagram, use case. You will include flow diagrams for your work. But I would recommend in

e.g. 

Top down diagrams are used for procedural programming languages.



Class/Inheritance diagrams are for object orientated code, you do not need to do this if you are using a procedural language

Try and reduce this diagram to it’s smallest component parts.

You may wish to reference some of the computational think in this section by refereeing to:

* Abstraction – what have you discarded?
* Thinking ahead – what problems/limitations can you foresee?
* Thinking concurrently – quickly dismiss any thought of using cores independently
* Reference the next section to thinking procedurally

A black background with white rectangles

Description automatically generated

There are a couple of features that I have abstracted from this structure diagram. I have not included any input validation as it will all be contained inside the various functions and subprograms, such as the login and signup page. The main machine learning algorithm is contained inside the different pages for the program and will execute only on that page. I also have discarded all of the aesthetic objectives since they are not relevant in the structure diagram.

Some of the immediate issues I can see is the database storage. If this will be done centrally, this will require lots of networking and cloud storage may have to be used. This essentially creates another problem in the structure diagram, so I may decide to not include cloud database storage depending on the development process. There are some other features that I may decide to change during the development process as well.

It will be hard to use cores independently as each screen in the program is self-contained, and hence cannot be worked on while other screens are running. This only leaves the main machine learning algorithm to run alongside the visual outputs, and the visual outputs do not require much computing power and hence concurrent processing will not be used in the program.

## (ii) DESCRIBE THE SOLUTION

*(a) Explain and justify the structure of the solution.*

*(b) Describe the parts of the solution using algorithms justifying how these algorithms form a complete solution to the problem.*

*(c) Describe usability features to be included in the solution.*

*(d) Identify key variables / data structures / classes justifying choices and any necessary validation.*

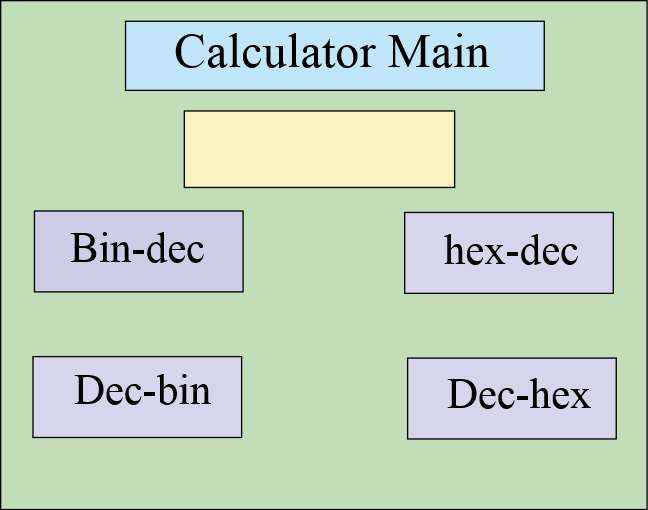
*(e) Identified and justified the test data to be used during the iterative development of the solution.*

Explain any connections, dependencies or links between parts of you design e.g. the main page will have a login drawing on data from users table.

Modularise you work e.g. complete these elements for each of your modules (screens)

Using the example above – the calculator

**Calculator – Main Page**

****

The main page will contain the options for my binary calculator. I will have four options that the user is allowed to choose (bin-den, den-bin, hex-den, den-hex) and they can enter the number onto the front page. It will also store their number in a file for later use. There is a title, data entry for number buttons……

Remember to list all of the variable/functions used, their purpose, data types and validtions

**Functions and variable used:**

Var click\_me Boolean 0 or 1 validation – lookup check 0 or 1

Var sci\_bin String (3) – validation – length check (3 chars)

Func\_conv\_bd(number) function that converts binary numbers to denary numbers

**Flow chart to show the process**

START

Bin to den?

Den to bin?

Remember to include routines for the validations

*……………………….etc*

**Pseudo code**

Proc calculator\_main

Open main\_page

If x < 7

Open random\_message(“this must be a binary number”

End if

………………………………….etc

In order to test the functionality of this page, I will test the following items

**Button1**

Should close the main page and open the denary – binary converter

**Button 2**

Should close the main page and open the binary – denary converter

………………………etc

You need to plan the tests that you will use as you are developing this module – it can be quite short – tests should be designed to demonstrate that your system is working – particularly validations and the logic of the module.

**Test Plan for development**

**For var\_number**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test data | Test type | Predicted outcome | Actual outcome | Pass/Fail |
| 010100101 | valid | Saves number |  |  |
| 01010100101001010100 | borderline | Saves number |  |  |
| 000000000000 | boarderline | Saves number |  |  |
| bob | Invalid | Rejects text |  |  |

………..etc

**Login/Sign up page**

**A screen shot of a login screen

Description automatically generated**

The login and sign-up page will consist of a title at the top, and the main colour scheme and structure of the page will be consistent across all screens. There are 2 input boxes, one for the username and one for the password. These variables will then be stored temporarily and checked/added against/to the existing logins database to see if a login/signup is successful. If so, the program will proceed to the main menu using the appropriate account. The input boxes contain placeholder text as well as labels to make it easier for the user to use. The login and signup buttons are underneath the input boxes. The login button will just check the credentials and login. If not found, a red error message will be displayed near the top of the screen as can be seen. This is also true for the sign-up page, but different validations will occur and a different error message will pop up onto the login page.

**Login**

**Variables used:**

* **var username**: String. Captures the user's input for the username. Required to be between 1 and 15 characters to prevent input boxes from being too full.
* **var password**: String. Captures the user's input for the password. Required to be between 1 and 15 characters to prevent input boxes from being too full.

All of the fields will be trimmed beforehand and stored, which will remove leading and trailing spaces

**Function ValidateInputs(username, password), returns Bool**

* Checks that the username is not empty and within the 1-15 length range. Displays the relevant error message in the adjacent error label
* Checks that the password is empty and within the 1-15 length range. Displays the relevant error message in the adjacent error label

**Function AuthenticateUser(username, password) returns Bool**

* Only runs if ValidateInputs is true
* Creates a SQL query that gets the number of records in the logins database useraccounts table with the specified username and password. The password is encrypted and then compared to the encrypted passwords in the database
* If the SQL query returns 0, then return false and display username and password combination incorrect error message
* If the SQL query returns a value larger than 0, then return true and proceed to the simulation page of the program

**Procedure ClearButtonClicked**

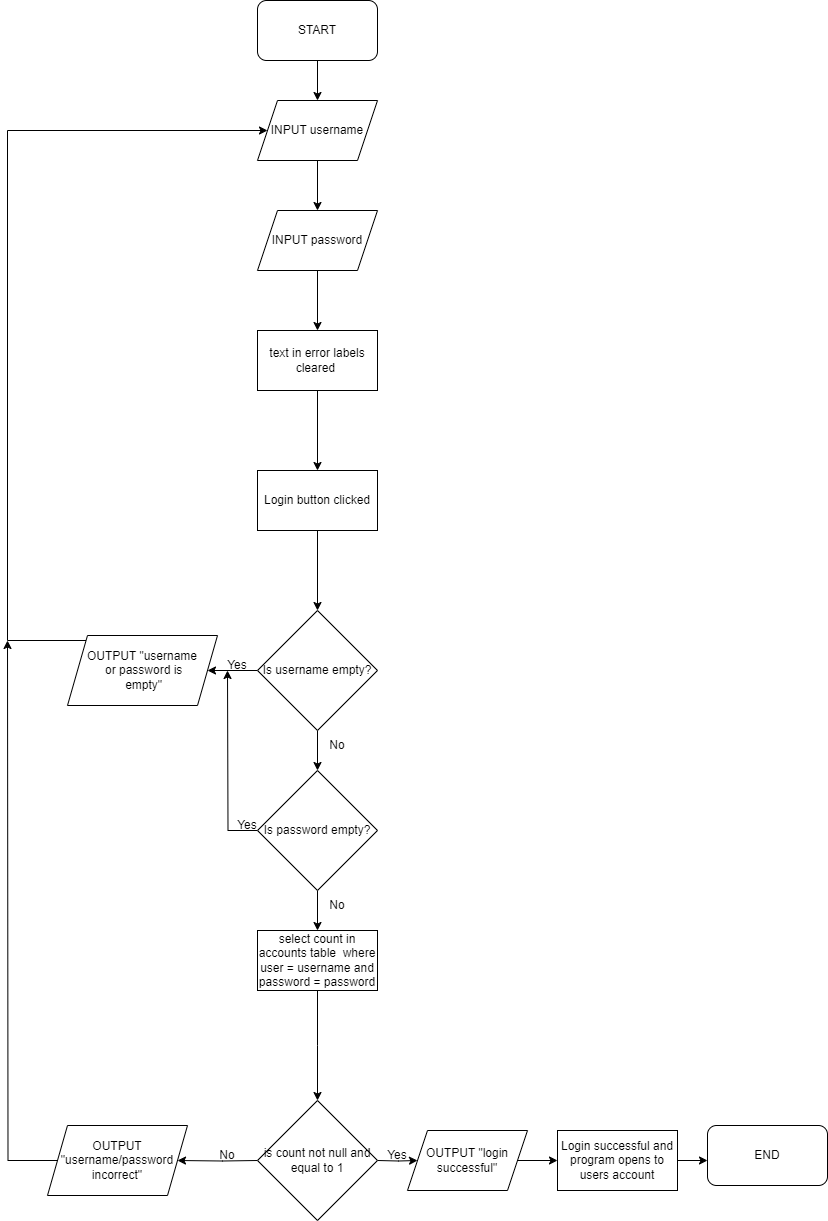
* Executes when clear button is clicked
* Clears values inside input boxes and error label
* Unchecks show password checkbox such that password appears as \*

**Procedure ShowPasswordCheckboxChanged**

* Executes when the show password is checked/unchecked
* This will hide the password as \* if the checkbox is unchecked and will show the password if the checkbox is checked

**Procedure LoginButtonClicked**

* This should run the functions ValidateInputs and RegisterNewUser, and should be disabled after being clicked once in a cycle



**Flowchart**

**Pseudocode**

**Client side**

Function ValidateInputs() Returns Boolean

isValid = True

If UserName is empty

Set UserName error label to "Username is required"

isValid = False

If Password is empty

Set Password error label to “Password is required”

isValid = False

Return isValid

**Server side**

Function AuthenticateUser(username, password) Returns Boolean

Connect to database

Define query for user authentication using USERNAME=username and PASSWORD=encrypted password

Execute query and store the result

If result is 1

Return True

Return False

Testing plan

**Username text input** - this should store the inputted value inside the variable username and should lie within the allowed data values. Placeholder text should be removed as typing starts.

**Password text input** - this should store the inputted value inside the variable password and should lie within the allowed data values. There will be no placeholder text.

**Procedure LoginButtonClicked** – this should run the ValidateInput and AuthenticateUser functions. It should be disabled after being clicked once in a cycle

**Procedure ClearButtonClicked**

* Executes when clear button is clicked
* Clears values inside input boxes and error label
* Unchecks show password checkbox such that password appears as \*

**Procedure ShowPasswordCheckboxChanged**

* Executes when the show password is checked/unchecked
* This will hide the password as \* if the checkbox is unchecked and will show the password if the checkbox is checked

**ValidateInputs**

Username

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test data | Test type | Predicted outcome | Actual outcome | Pass/Fail |
| “user” | valid | Error message not visible |  |  |
| “” | invalid | Displays empty error message |  |  |
| “ “ | invalid | Displays empty error message |  |  |
| “a” | borderline | Error message not visible |  |  |
| “username user21” | borderline | Error message not visible |  |  |
| “usernameusernam” | invalid | Should not be able to type into input box |  |  |
| “ user program ” | valid | Error message not visible |  |  |

Password

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test data | Test type | Predicted outcome | Actual outcome | Pass/Fail |
| “user” | valid | Error message not visible |  |  |
| “” | invalid | Displays empty error message |  |  |
| “ “ | invalid | Displays empty error message |  |  |
| “a” | borderline | Error message not visible |  |  |
| “username user21” | borderline | Error message not visible |  |  |
| “usernameusernam” | invalid | Should not be able to type into input box |  |  |
| “ user program ” | valid | Error message not visible |  |  |

The validations for password and username on the client side for the input box itself are the same so I have included the same test cases for both username and password input boxes.

**func AuthenticateUser**

Assuming we have [[“user1”, “password1’], [“user2”, “password2”]] as our password username pair in a logins database, we can construct a test plan that will test certain username password pairs.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test data | Test type | Predicted outcome | Actual outcome | Pass/Fail |
| “user1”, “password1” | valid | Error messages not visible |  |  |
| “user1”, “password2” | invalid | Displays not authenticated error message |  |  |
| “user2”, “password2” | valid | Error messages not visible |  |  |
| “user1”, “user1” | Invalid | Displays not authenticated error message |  |  |
| “user14”, “password1” | invalid | Displays not authenticated error message |  |  |

**Register**

**Variables Used**

* **var username**: String. Captures the user's input for the username. Required to be between 1 and 15 characters to prevent input boxes from being too full.
* **var password**: String. Captures the user's input for the password. Required to be between 1 and 15 characters to prevent input boxes from being too full.
* **var emailAddress**: String. Captures the user's input for the email address, validated against a standard email format, max length is 30.
* **var confirmPassword**: String. Captures the user's input for confirming the password, compared with the password variable for equality.

All of the fields will be trimmed beforehand and stored, which will remove leading and trailing spaces.

**Functions and Logic Used**

**func ValidateInputs() returns Bool**

* **Username Validation**: Ensures the username is non-empty and within the 1-15 character range. Displays an adjacent error message for invalid inputs.
* **Email Address Validation**:
  + Verifies non-emptiness of the email address.
  + Uses regex ("^[^@\s]+@[^@\s]+\.[^@\s]+$") to validate the email format, displaying an error for mismatches. Regex taken from https://emaillistvalidation.com/blog/email-validation-in-javascript-using-regular-expressions-the-ultimate-guide/
* **Password Validation**:
  + Confirms the password's non-emptiness and adherence to the 8-15 character range.
  + Applies regex (@"^(?=.\*\d)(?=.\*[!@#$%^&\*])(?=.\*[a-zA-Z]).{8,15}$") to ensure it contains at least 8 characters, 1 number, and 1 special character, indicating invalidity through an error message otherwise. This regex has been taken from <https://stackoverflow.com/questions/19605150/regex-for-password-must-contain-at-least-eight-characters-at-least-one-number-a> and adapted to meet requirements.
* **Confirm Password Validation**: Checks for a match with the password, indicating discrepancies through an error message.

**Procedure RegisterNewUser(server\_successful, client\_successful)**

* Checks if username/email used with registration is already in logins database
* If so, relevant error message will be displayed
* Otherwise, it will add username, email address and password as a record to the database

**Procedure ClearButtonClicked**

* Executes when clear button is clicked
* Clears values inside input boxes and error label
* Unchecks show password checkbox such that password appears as \*

**Procedure ShowPasswordCheckboxChanged**

* Executes when the show password is checked/unchecked
* This will hide the password as \* if the checkbox is unchecked and will show the password if the checkbox is checked

**Procedure RegisterButtonClicked**

* This should run the functions ValidateInputs and IsSignUp, and should be disabled after being clicked once in a cycle

A black background with white squares

Description automatically generatedFlowchart

Pseudocode

**Client side**

Function ValidateInputs() Returns Boolean

Set allValid to True

Reset error labels for UserName, Password, EmailAddress, and ConfirmPassword to empty

If UserName input is not valid

Set UserName error label to "Username is required"

Set allValid to False

If EmailAddress input is not valid

Set EmailAddress error label to "Email address is required"

Set allValid to False

Else

Define pattern for valid email

If EmailAddress does not match the pattern

Set EmailAddress error label to "Invalid email address"

Set allValid to False

If Password input is not valid

Set Password error label to "Password is required"

Set allValid to False

Else

Define pattern for strong password

If Password does not match the pattern

Set Password error label to "Password too weak"

Set allValid to False

If Password does not match ConfirmPassword

Set ConfirmPassword error label to "Password does not match"

Set allValid to False

Return allValid

Function IsValidInput(input) Returns Boolean

If input is null or empty

Return False

Return input is not just whitespace

**Server side**

Function RegisterNewUser(username, password, email) Returns Boolean

Open connection to database

Define query to check existence of USERNAME or EMAIL

Execute query and store the result

If username/email already in database`

Set username/email error label to username/email unavailable

Return false

else

Define query to insert new user with username, encrypted password, and email

Execute insertion query

Output Registration successful message

Return true

A combination of these functions will be used to build the final solution for registration. Once RegisterNewUser has returned true, the program will proceed to the simulation page.

Test plan

**Username text input** - this should store the inputted value inside the variable username and should lie within the allowed data values. Placeholder text should be removed as typing starts.

**Password text input** - this should store the inputted value inside the variable password and should lie within the allowed data values. Placeholder text should be removed as typing starts and will contain the strength requirements for the password

**Email text input** - this should store the inputted value inside the variable email and should lie within the allowed data values. Placeholder text should be removed as typing starts.

**Confirm password text input** – this should store the u=inpuuted value inside confirmPassword and will have not have as much input validation as the only validation is checking whether it equals password. There will be no placeholder text

**Button Signup** - this should run the ValidateInputs function using username and password as parameters, and the output of this function will run the IsSignup function. It should be disabled in a cycle if it has been clicked once already.

**Button Clear** – this should clear all input fields and uncheck the show password checkbox to hide the password again

**Show password checkbox** – this should show the password when checked and hide when unchecked

The last 2 functions should not need a test plan because this can immediately be verified without a test plan, and creating a test plan would not be an efficient use of time in project.

ValidateInputs()

Username

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test data | Test type | Predicted outcome | Actual outcome | Pass/Fail |
| “user” | valid | Error message not visible |  |  |
| “” | invalid | Displays empty error message |  |  |
| “ “ | invalid | Displays empty error message |  |  |
| “a” | borderline | Error message not visible |  |  |
| “username user21” | borderline | Error message not visible |  |  |
| “usernameusernam” | invalid | Should not be able to type into input box |  |  |
| “ user program ” | valid | Error message not visible |  |  |

Password

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test data | Test type | Predicted outcome | Actual outcome | Pass/Fail |
| “pass” | invalid | Displays weak password error message |  |  |
| “” | invalid | Displays empty error message |  |  |
| “ “ | invalid | Displays empty error message |  |  |
| “password123” | invalid | Displays weak password error message |  |  |
| “password@1” | invalid | Displays weak password error message |  |  |
| “passwo1@” | borderline | Error message not visible |  |  |
| “password123456@” | borderline | Error message not visible |  |  |
| “password123456@1” | invalid | Should not be able to type into input box |  |  |
| “pass123@” | invalid | Displays weak password error message |  |  |

Email

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test data | Test type | Predicted outcome | Actual outcome | Pass/Fail |
| “bob” | invalid | Displays wrong format error message |  |  |
| “” | invalid | Displays empty error message |  |  |
| “ “ | invalid | Displays empty error message |  |  |
| “sri@gmail” | Invalid | Displays wrong format error message |  |  |
| “srigmail.com” | invalid | Displays wrong format error message |  |  |
| “ben@gmail.com$” | invalid | Displays wrong format error message |  |  |
| “sri@gmail.com” | valid | Error message not visible |  |  |
| “sri123@patesgs.org” | valid | Error message not visible |  |  |

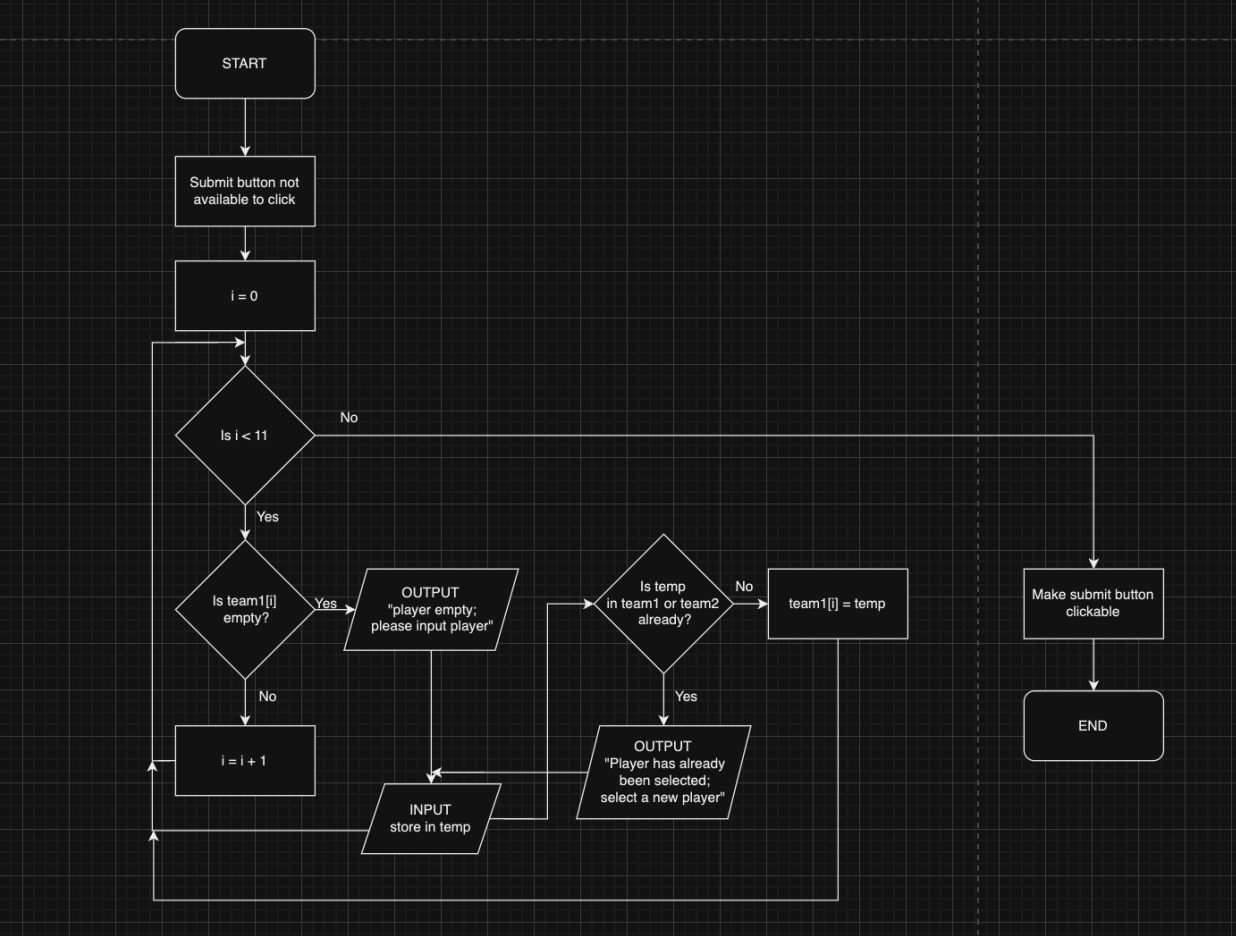
Confirm password

Confirm password is very simple to test, and only requires 2 test cases, one that matches the password and one that doesn’t. If it doesn’t match, the error message “mismatch” should be displayed. Otherwise, no error message is displayed.

func RegisterNewUser

Assuming we have [[“user1”, “password1@”, “user1@gmail.com”], [“user2”, “password2@”, “user2@gmail.com”]] as our records in a logins database, we can construct a test plan that will test certain username password pairs. All of these assume that password and username are valid, since these will be server side functions.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test data | Test type | Predicted outcome | Actual outcome | Pass/Fail |
| “user1”, “password123, “user1@gmail.com” | invalid | Displays duplicate username and email error message |  |  |
| “user2”, “password22@”, “user3@gmail.com” | invalid | Displays duplicate username error message only |  |  |
| “user3”, “password2@”, “user2@gmail.com” | invalid | Displays duplicate email error message only |  |  |
| “user4”, “pass2345@”, “user4@gmail.com” | valid | Returns true |  |  |

****

## (iii) DESCRIBE THE APPROACH TO TESTING

1. *Identify the test data to be used during the iterative development and post development phases and justify the choice of this test data.*

This is the overall testing for the system and should be a test plan that proves that you have completed the system and met all of your success criteria. You need to describe your methods of testing – these could include alpha, beta, white box, black box, top down & bottom up. There are several ways in this should be achieved:

1. System test – derive three scenarios that need to be completed, e.g. a customer needs their details entered, they wish to make a purchase and get their receipt. Choose one to be valid, the next to be borderline and the last to have invalid data in it.
2. Ask you user to test your system. Give them two tasks to do so that they can experience the usability – acceptance testing (black box)
3. White box testing – stress test variables & functions to try and expose weaknesses, e.g.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test data | Test type | Predicted outcome | Actual outcome | Pass/Fail |
| 010100101 | valid | Saves number |  |  |
| 01010100101001010100 | borderline | Saves number |  |  |
| 000000000000 | borderline | Saves number |  |  |
| bob | Invalid | Rejects text |  |  |

# (3) DEVELOPING THE SOLUTION (25 mARKS)

## (i) INTERATIVE DEVELOPMENT PROCESS

*(a) Provide annotated evidence of each stage of the iterative development process justifying any decision made.*

*(b) Provide annotated evidence of prototype solutions justifying any decision made*.

Creating the Program

I initially started by setting up Visual Studio as my IDE, and I installed the relevant features on it, like Git. With my program, I am going to build the UI and the main machine learning algorithm independently and then combine the builds into my final solution.

First, I started out by creating the project, and assigned the solution to be in the same folder as the code. This makes running the code a lot easier as I do not have to navigate between folders.

A screenshot of a computer

Description automatically generated

When a project, is created, several files are created as well to go alongside it. The main files are the program c# files and the form files. A form is a screen inside a project, and I will need to create separate forms for each screen in my final project.

I will be starting with the login page for my program.

A computer screen with a white square on it

Description automatically generated

This is the main interface for developing the program, and this is the designer preview that you can use to add different tools and functionality.

A screenshot of a computer program

Description automatically generated

I have renamed the forms to make them easier to recognise and work with, from form1 to LoginForm.

I will start by adding the title bar for the program to change it from form1 to Login, and defining some other form properties such that the form is always maximised, and it is not resizable.

A computer screen shot of a program

Description automatically generated

This is the automatically generated code that alters the state of the form. The use of maximizeBox set to false means that resizing is not possible, and setting the text to Login means that the title bar will now display Login. ClientSize is the size of the part of the form where controls can be added, and must be determined at a later point through experimentation.

A screenshot of a desktop

Description automatically generated

Here you can see the maximise button being disabled, and the title is changed to Login. This shows that the code is working as the form also opens to max size. I have added a label to create the title for my form.

A screen shot of a computer

Description automatically generated

I have changed the label’s properties to make sure that it meets my requirements that I have made. The size and anchoring will need to be adjusted later.

A computer screen shot of a program

Description automatically generated

This is the autogenerated code for the label. I have set the colour to blue, and have set the text to align in the middle of the label, horizontally and vertically. I have set its location to 0,0 as it is the most effective place for the anchor to be, as it needs to be at the top of the page. The anchor is set to top, left and right so that it remains at the top. I next need to add the size properties for the label to ensure that it covers the entire screen horizontally.

I did this by assigning autoresize to off and adding a dock to the top, such that the label sticks to the top of the form.

## (ii) TESTING TO INFORM DEVELOPMENT

*(a) Provide annotated evidence for testing at each stage justifying the reason for the test.*

*(b) Provide annotated evidence of any remedial actions taken justifying the decision made.*

# (4) EVALUATION (20 mARKS)

## (I) TESTING TO INFORM EVALUATION

*(a) Provide annotated evidence of testing the solution of robustness at the end of the development process.*

*(b) Provide annotated evidence of usability testing (user feedback).*

## (II) SUCCESS OF THE SOLUTION

*(a) Use the test evidence from the development and post development process to evaluate the solution against the success criteria from the analysis.*

## (III) DESCRIBE THE FINAL PRODUCT

*(a) Provide annotated evidence of the usability features from the design, commenting on their effectiveness.*

## (IV) MAINTENANCE AND DEVELOPMENT

(*a) Discuss the maintainability of the solution.*

*(b) Discuss potential further development of the solution.*

## APPENDIX A - BIBLIOGRAPHY