A2 Unit 4 Computing Project

Football Club Membership System

Hannah Short

Contents

Analysis

Client: Burgess Hill Town Football Club

Contact: Emily Hodgkinson

The Green Elephants Stadium
Leylands Park
Maple Drive
Burgess Hill
West Sussex
RH15 8DL
(01444 254832)

Identification of the Problem:

Burgess Hill Town FC is a small football club with the number of people attending matches having increased significantly over the past few years. Currently, most of their operations are paper-based and their website lacks interactivity with their supporters. At present its only function is storing primary information about the club.

The Current System:

Currently all supporters purchase tickets manually over the counter at the entrance of the club, this can mean on particularly busy matches, queues tend to build up quite quickly. Tickets can only be purchased with cash by both its members and standard supporters. Members hold a membership card displaying their basic details and which when shown on entry enables them to get in at a reduced rate whilst others pay a fixed price. At the moment, the only means of contacting the club regarding match information is via the phone number displayed on the club website making it difficult for the staff if there are multiple queries from site users at any one time. Furthermore, the names of the other teams in the league of which Burgess Hill will play in the season are shown on the website's fixture page; however there is no club location or postcode given with the team name making it harder for the supporters to get to away games.

Interview:

1. How does the current ticket system operate?

    A. At the club, customers always queue through turnstiles to pay for a ticket/tickets to get into the game on match day

2. How does the pricing of tickets differ?

    A. Its £9 for an adult ticket, £5 for members and OAPs and free entry for people under 18

3. What are the advantages of this current system?

    A. Supporters don't need to pre-book tickets; they just turn up on match day and buy one. Also, people of all age ranges can come and pay to see the games and selling over the counter allows us to see the members' membership cards to check they're genuine and belong to the right people

4. Is there a limit to how many tickets you can sell?

    A. No, although there are approximately 200 seats available, there is plenty of room for people to stand round the edge of the pitch. Usually, the people who are first to buy their tickets get seats in the stand

5. What are the drawbacks of the ticket system?

    A. As it's based on who turns up on the day, it can be hard sometimes to predict what turnout we'll get. In some situations, this has a negative financial impact on the club because it's possible we end up selling very few tickets. On the other hand, when a game has a large turnout, it often takes a lot of time to get everyone through the turnstiles and into the game as a result of long queues.

6. In what ways would it help having a computerised format of this system?

    A. It would help if everyone who regularly came to games had an electronic membership account on an online ticket booking system for the club as it would mean the majority of supporters would be able to get into games straight away having already bought a ticket and it would give the club a rough estimate of how many people were coming.

7. How would you want the club members to purchase their tickets?

    A. It would be good if the members had a personal login where they were able to enter their membership number and a password of their choice so they could select the tickets they wanted to get

8. What payment method would they use?

    A. Something such as PayPal with a credit or debit card where we could keep a record of the member's card details

9. What content do you currently upload on your website?

    A. On the website, we put the fixtures and results on. Information and news regarding the club is also shown on there

10. Is the phone number the only method of contact you have with customers?

    A. Yes unfortunately. There is an email address for people to contact who are looking to book out the clubhouse for events but there is no means of communication for contacting us for team information other than the club phone number

11. How else would you like them to be able to contact the staff?

    A. If there was a method of e-mailing us on the website that people could use to send any queries and questions they had to an e-mail address, it would be beneficial as we could respond to them as and when unlike phone calls which can only be answered one at any time

12. What is the most common thing people contact you about?

    A. A lot of the time we have people contacting us asking for the locations of away games and directions to them as we don't upload any details other than the name of the club we are going to play on our website

13. How do you think this situation could be improved?

    A. If the postcode of the club we were going to was provided and a possible route with directions to the grounds of the other clubs

14. Is it generally members that attend away matches?

    A. Yes, normally

15. In your opinion, what would be the best way to display such a solution with user logins?

    A. In a separate section accessed via the website for members to access things such as a booking system, a space for contacting the club and directions to away games

Members Survey:

After interviewing somebody from the club website staff, I surveyed a current member of the football club to assess the types of problems they face with the current system and club website. From the responses, I will evaluate which benefits of computerising the club operations they would find most useful, new features for possible development in a new system and how they felt the system should be presented.

What are the biggest drawbacks of the current football club operations and website?

Long queuing times ☐

Bringing money to the ground ☐

Lack of club communication with members and supporters ☐

Accessing away games ☐

Lack of information on club website ☐

In a computerised version of the current system, which aspects would be most useful?

Not needing to bring money to the ground ☐

Paying for tickets in advance ☐

Having information stored electronically in event of queries ☐

Avoiding queues into ground ☐

What new features would be useful for the club to implement?

Personal accounts for members ☐

Displaying up-to-date timings of all league games ☐

Giving users routes to away grounds ☐

Advertising memberships to standard supporters online ☐

Improved method of contact with club ☐

What would be the most effective means of accessing a computerised system?

A new club website ☐

A second website running alongside the original for new features ☐

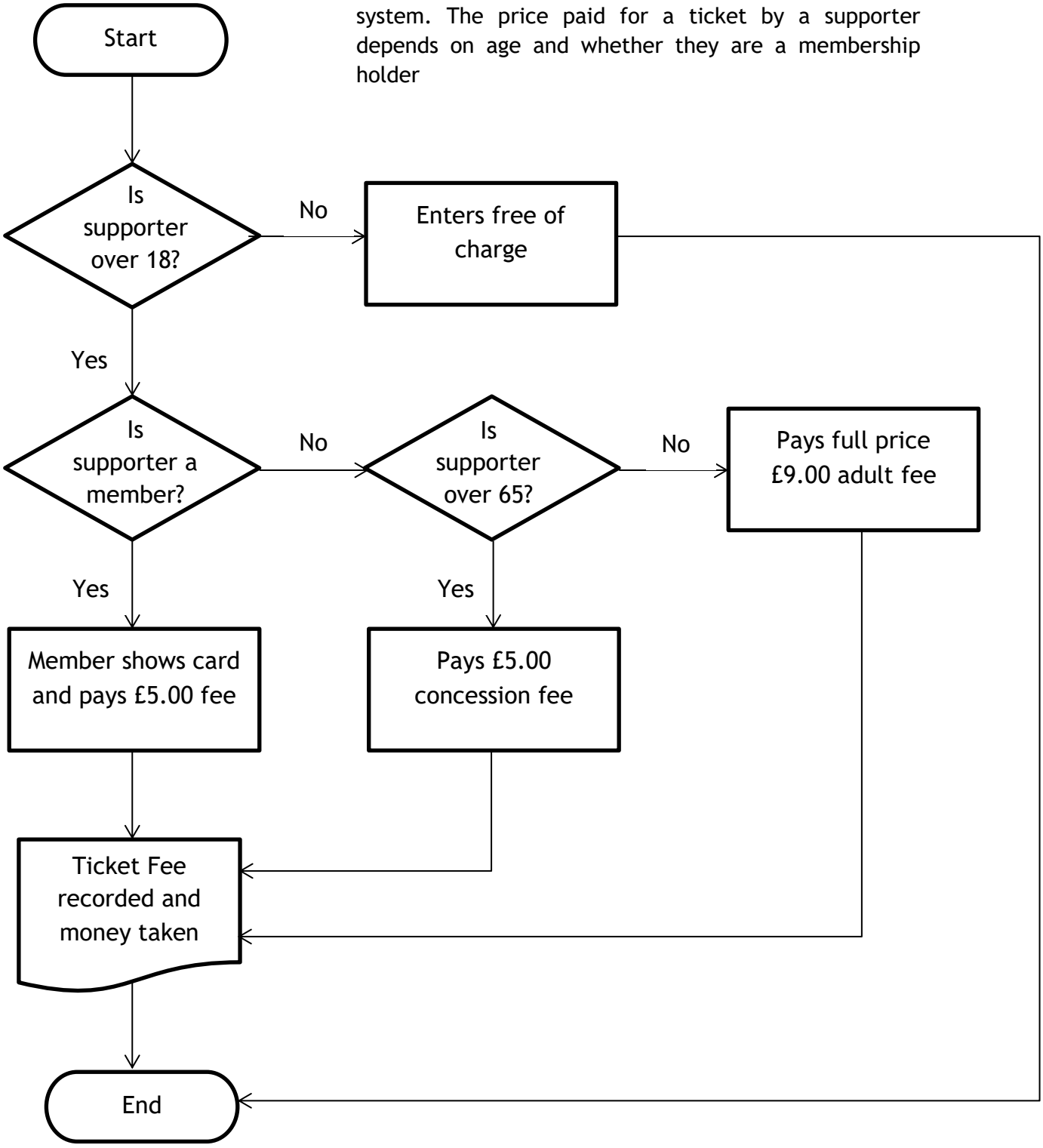
A system accessed from the current website ☐

A special-purpose program for the club ☐

Other comments:

System Flowchart:

This flowchart represents the current ticket sales system. The price paid for a ticket by a supporter depends on age and whether they are a membership holder



Prospective Users:

The primary users of this system would be the members of the club to book tickets to the matches, plan their routes for going to away matches and having an improved form of contact with the club. They need to find the system safe and easy to use with a clear understanding of how the different sections function.

However the system will also need to be used by the club personnel to look up member's details, check payments and respond to queries from the e-mail account. The website staff will be ultimately responsible for the running of the system but the matchday staff will be required to use it also to ensure all members have paid and look up any details a member may require. The level of computer skill of the matchday staff is unknown so the system needs to be as straightforward as possible and function as quickly if not quicker than the current system.

User Needs and Acceptable Limitations:

User Needs:

- The user needs an easy to use membership system for the football club whereby it is quick, safe and easy to pay for match tickets
- Each member of the club needs a personal account where they are able to log in with their membership number and a password of their choice
- The members need to be able to select the types of tickets they need and how many they need before the system calculates the total cost
- The members need to be directed to a PayPal page where the transaction can be processed
- Once tickets have been purchased, the member needs to be provided with a ticket booking confirmation
- The system needs to provide an area where new members can join and their information is stored by the club
- The member needs access to a route planner for away matches where they are advised on the best route to be taken to an away match
- The address and postcode of the ground of the away team needs to be clearly displayed to the member
- The member needs to be able to contact the club via an e-mail box on the system where they can type short messages for the club to respond to
- The user needs to be able to lookup information regarding the members of the club such as their personal details and the tickets they've purchased

System Limitations:

- Time limitations = The system needs to be completed by May
- Personnel knowledge limitations = The solution and the user manual for the solution can't be too complex because they need to be understood by personnel who may not normally be familiar with the software
- Hardware limitations = the system may not work on all devices. For example, a member wanting to purchase tickets on their tablet might find the system incompatible with their device
- Software limitations = the club can only run a limited amount of software on desktop computers
- Data limitations = there will be a limit to how many members the database can store, this will also depend on the available memory on the computer hard disk

Data Sources and Destinations:

Current System:

Data	Source	Destination
Name	Member form	Membership Card
Membership Number	Generated by club	Membership Card
Members Address	Member form	Membership Card
Ticket type	Member	Record for club treasurer
Ticket quantity	Member	Record for club treasurer
Money paid	Cash from member	Record for club treasurer

Proposed System:

Data	Source	Destination
Name	Entered as new member	Stored in database
Membership Number	From membership card/randomly generated	Stored in database
Password	Entered as new member	Stored In database
Members Address	Entered as new member	Stored in database
Members Postcode	Entered as new member	Stored in database
Ticket type	Selected from drop down menu on form	Ticket confirmation/stored in database
Ticket Price	Club Prices	Used in calculating total ticket costs
Ticket quantity	Selected from drop down menu on form	Ticket confirmation/stored in database
Money paid	Calculated from ticket prices and quantity and debited from members bank account	Paid to club account
Card Type	Members debit/credit card	Entered on payment
Card Number	Members debit/credit card	Entered on payment
Card Expiry Date	Members debit/credit card	Entered on payment
Card Security Code	Members debit/credit card	Entered on payment
Match Date	Ryman website	Printed on ticket
Email Address	Members email account	Listed as sender in club's inbox
Phone Number	Entered by member	Sent to club in e-mail
Email Subject	Created by member	Name of message in club's inbox
Email Message	Created by member	Sent to club's email inbox
Away Team	Ryman website	Menu on route planner
Away Team Ground	AA maps	External file and then opened on route planner
Away Team Address	AA maps	External file and then opened on route planner
Away Team Postcode	AA maps	External file and then opened on route planner
Distance to Away Ground	AA maps	External file and then opened on route planner
Directions to Away Ground	AA maps	External file and then opened on route planner

Data Volumes:

- The system needs to be able to load successfully with all its data when launched by the user
- The system needs a screen displaying the options available to the member when they log on which can navigate to various other screen depending on the option selected, the system needs to be able to process these navigations
- The system needs to store a membership number and password unique to each member and will need to be able to generate membership numbers for new members joining and store these also
- The system needs to process the membership number and password when the member logs on and compare them to the already existing ones to allow the user to log on
- The system needs to be able to store the personal details of all the current members of the club and needs to be able to store details of all new members joining the club
- The system needs to cater for all ticket categories and quantities to calculate costs and store the value of the amount of money that has been paid by the member
- The system needs to store the names and locations of the away grounds in the Ryman youth league, this needs to eventually be extended to other leagues such as the first team and women's
- The system needs to be able to process e-mail messages and send them to a club address

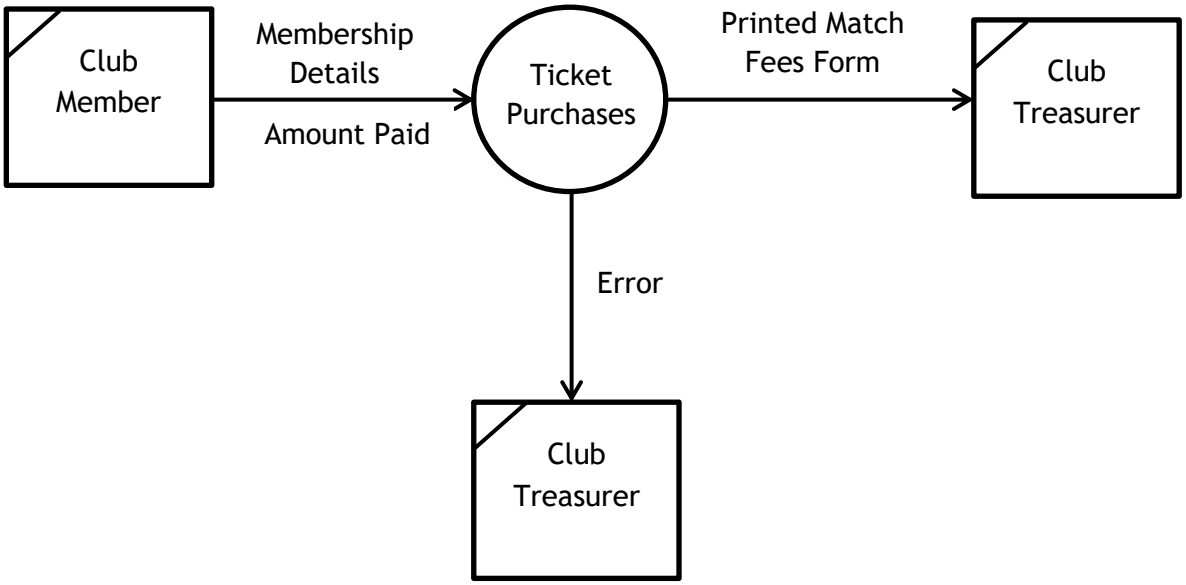
Analysis Data Dictionary:

Name	Data Type	Purpose/Description	Size/Range	Example Data	Validation
First Name	String	Stores name	-	John	-
Surname	String	Stores name	-	Smith	-
Address	String	Stores Address	-	21 Sample Lane Burgess Hill West Sussex RH15 1AB	-
Membership No.	Integer	Identifies member	8	12345678	8 digit number on membership card
Ticket Type	String	Defines ticket purchased		Member	Lookup – Under 18, Member, OAP, Adult
Ticket Price	Currency	Stores price of ticket	£-.00	£5.00	Relates to Ticket Type
Quantity of Ticket Purchased	Integer	Stores number of particular ticket type bought	≥1	2	
Total Amount Paid	Currency	Shows amount paid	£-.00	£5.00	Check cash given is equal to total amount paid
Date Paid	Date/Time	Stores date of match where tickets were bought	NN/NN/NN	01/01/15	Ensure date is in valid format

Data Flow Diagrams:

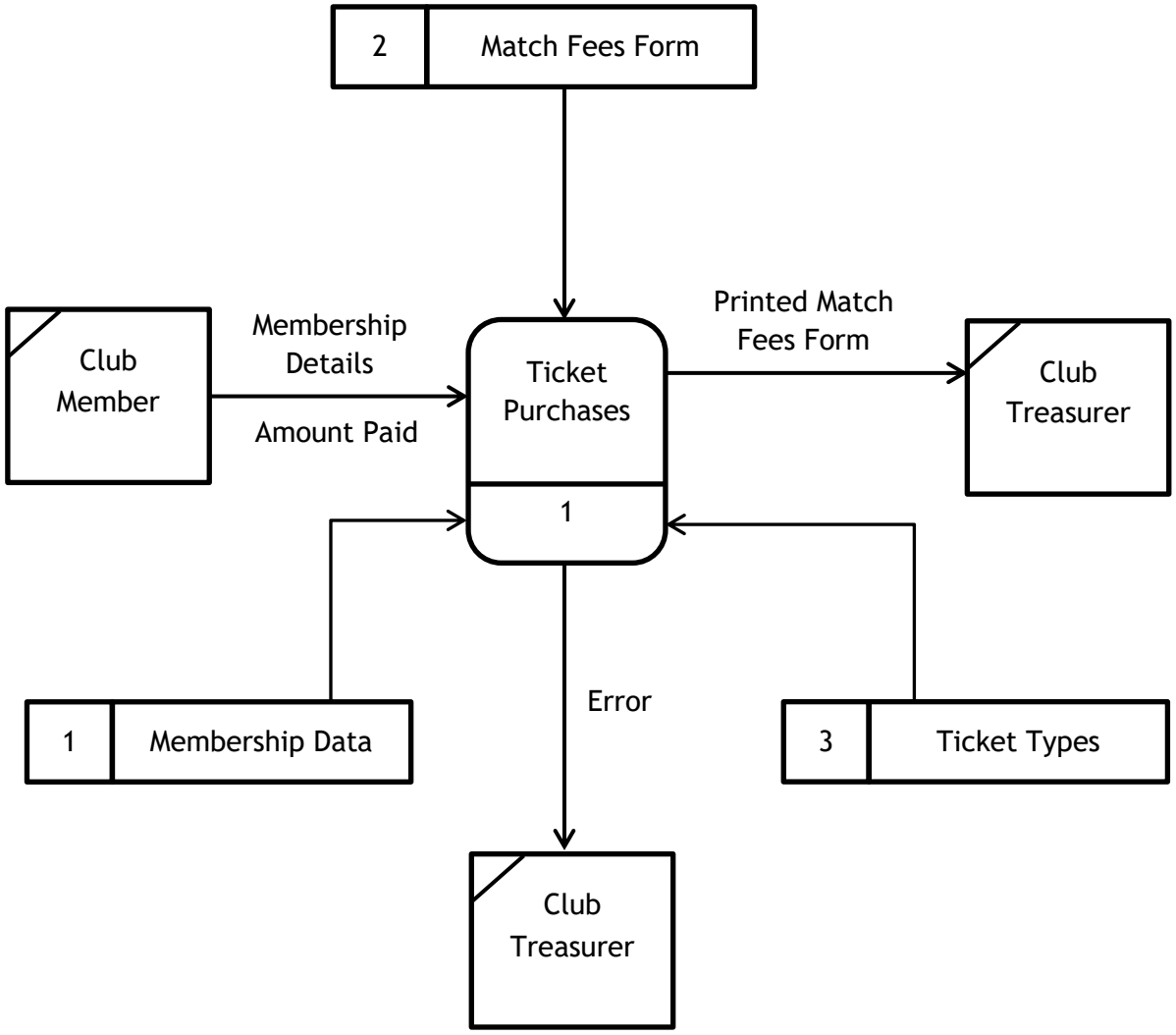
Existing System:

Context Diagram:



This context diagram represents how the data flows in the current system. At the moment it is very basic. The data source is the club member who gives their details via their membership card and selects the types and quantities of tickets they want to purchase. This is then manually processed by the matchday staff which give them the tickets in return. All the data is recorded on paper and then passed on to the club treasurer to calculate revenue.

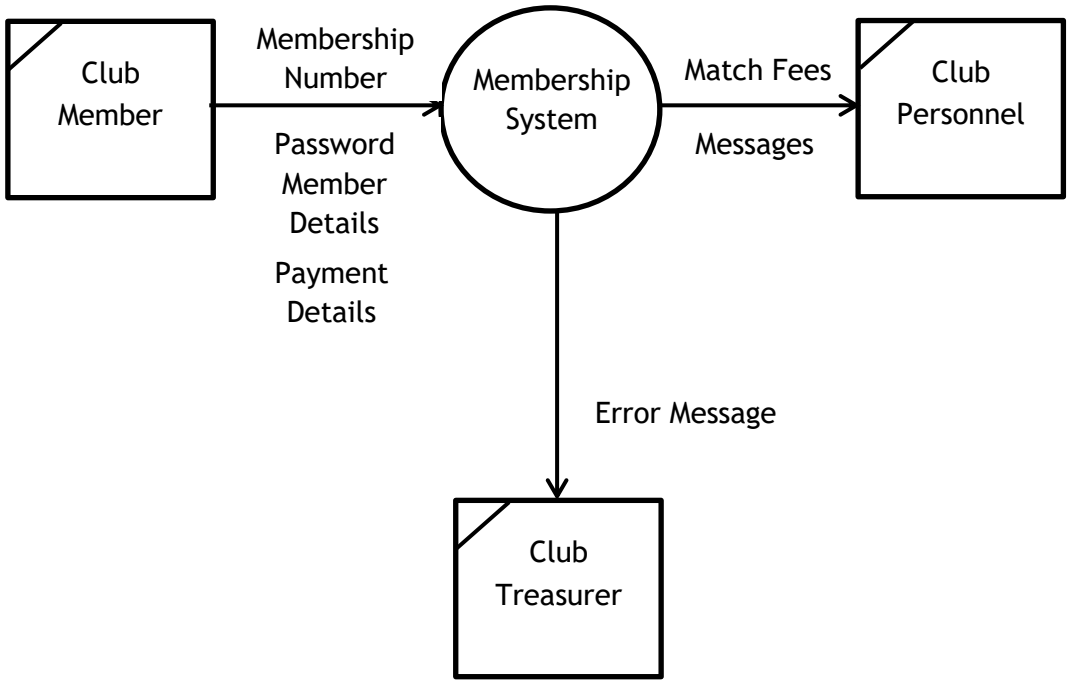
Level 1 Data Flow Diagram:



This lower level diagram shows the data flow that occurs in the current system in slightly more detail. It shows the data stores that are inputted into the system and the subsequent data outputted.

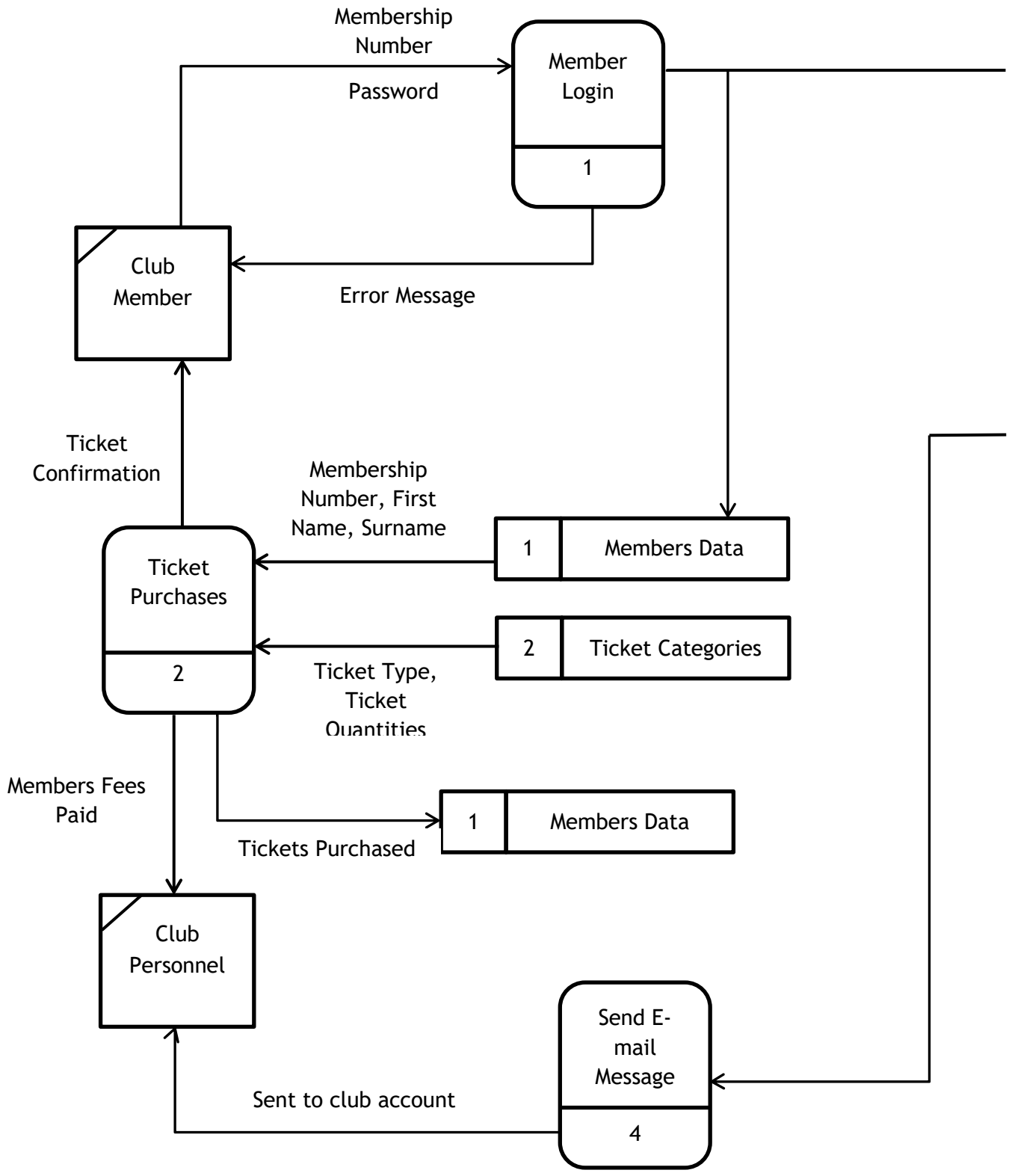
Proposed System:

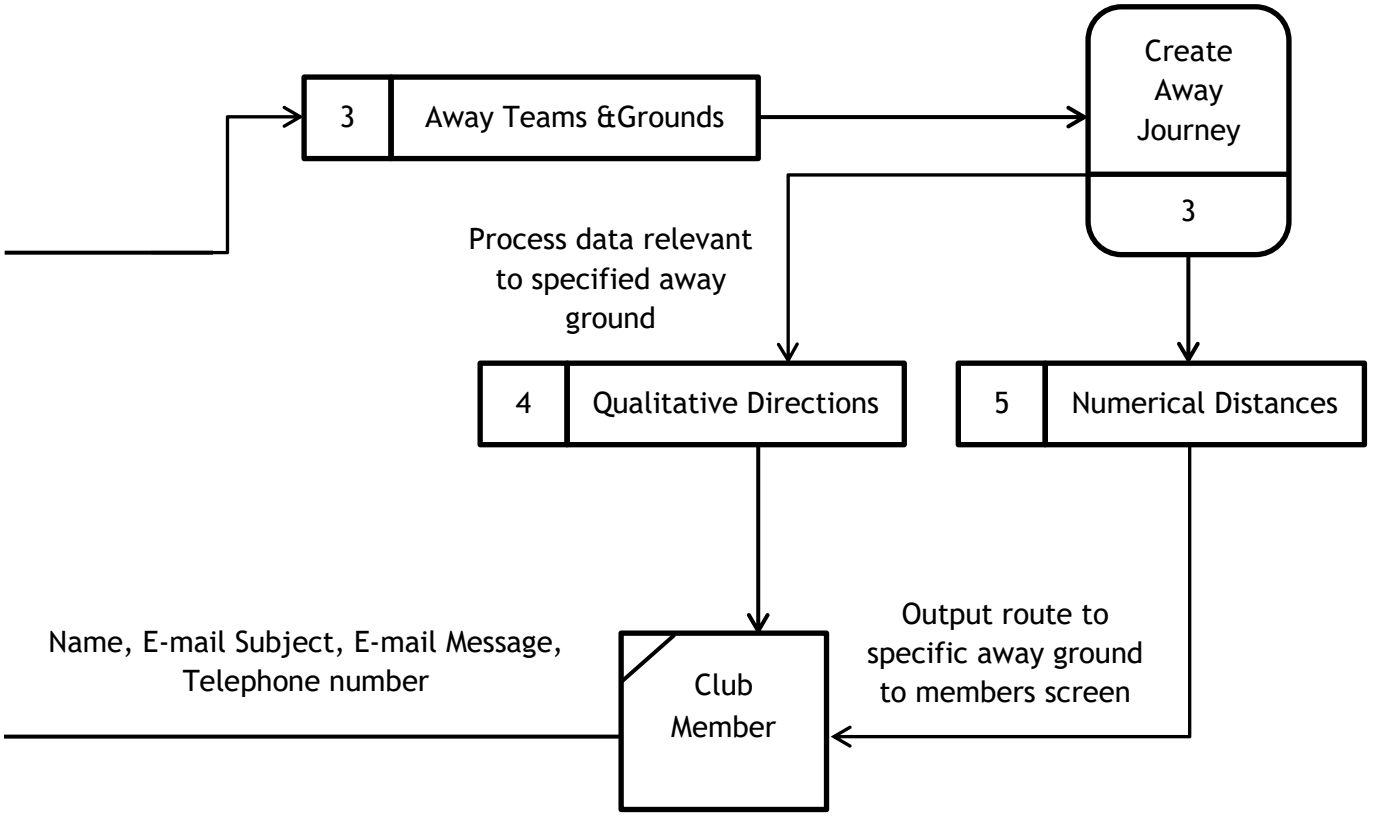
Context Diagram:

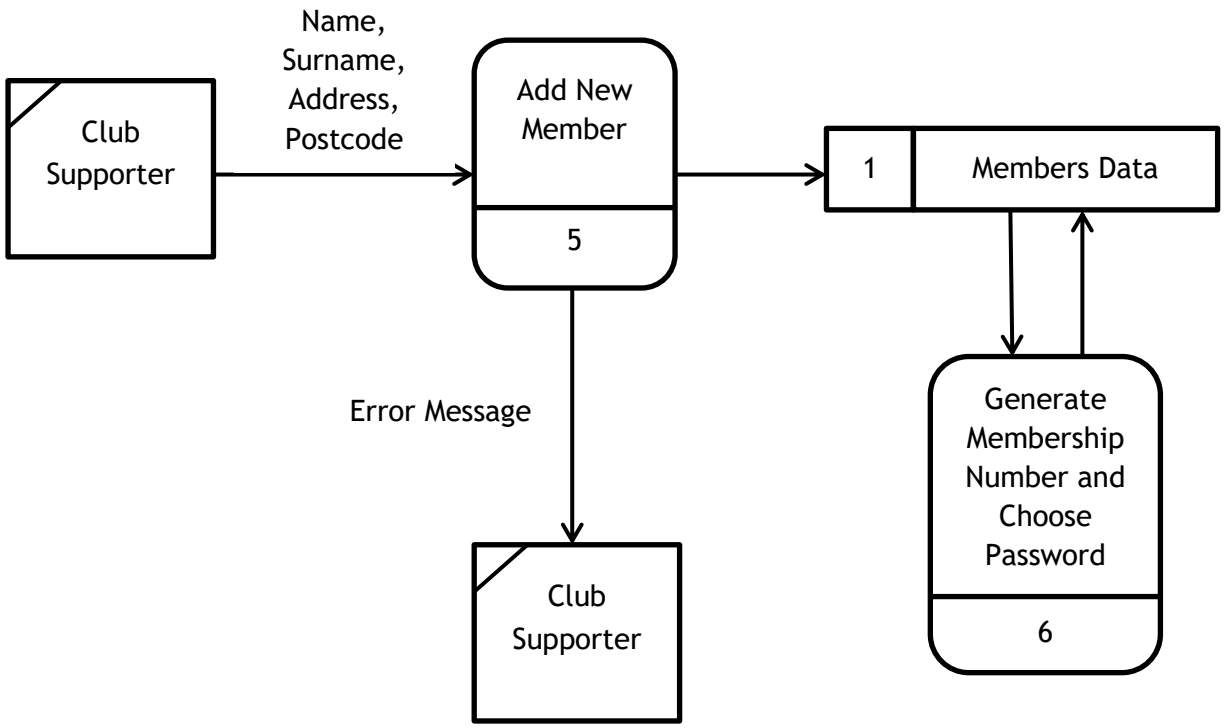


This high level diagram shows how the data will be moving in the new system at an abstract level. The member will gain entry to the club by purchasing tickets to a match on the system in advance of the game. They will be able to pay by card and their account will be debited with the according fee. On matchday, the member will need to show the ticket or proof of purchase

Data Flow Diagram:




3
Away Teams &Grounds
Create Away Journey
3
Process data relevant to specified away ground
4
Qualitative Directions
5
Numerical Distances
Name, E-mail Subject, E-mail Message, Telephone number
Club Member
Output route to specific away ground to members screen


Club Supporter
Name, Surname, Address, Postcode
Add New Member
5
1
Members Data
Error Message
Club Supporter
Generate Membership Number and Choose Password
6

Description of proposed system data flow diagram:

In the proposed system, members will log on with their membership number and personal password. The system will check that the membership number is valid and that the password corresponds to the membership number entered by querying the member data that is stored. After going through a series of validation checks, an error message will be presented to the user if the data entered is not valid. Otherwise, they are given access to the system.

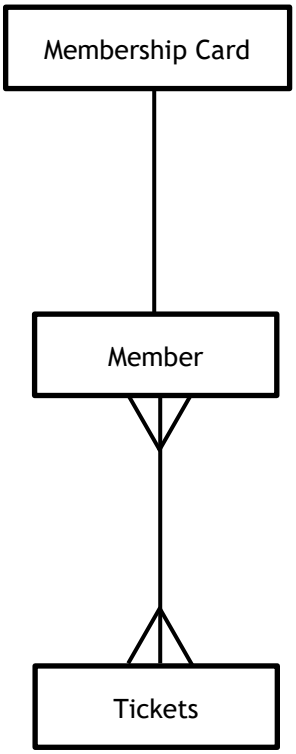
After logging on, the user can purchase their match tickets. They select their ticket types and ticket quantities from the categories available and the total cost of the purchase is calculated from the data chosen. When the tickets have been bought, the information about the tickets purchased by the member will be stored for future reference and a ticket confirmation will be available for the member.

Furthermore, the system will have access to data regarding away teams and their respective grounds so members can plan a route to away matches for the club. When the team has been selected, qualitative data for directions and quantitative data for distances can be accessed and the relevant data will be outputted to the screen for the member. Users will also be able to send e-mails directly to the club personnel by entering their name and phone number followed by an e-mail subject and the message.

If a standard club supporter wants club membership, they can enter their basic details and if validated, these will be added to the store of members' data. An error message will be returned if the new data they have entered isn't of a valid form, for example if the postcode isn't of the correct format. When their basic data has been added to the data store, they will have a membership number generated for them and will be able to choose a personal password for their account. After this, they will be able to log on to the system in the future.

Entity-Relationship Diagrams:

This is an entity-relationship diagram representing the current system:



Every member possesses their own unique membership card with their basic details on it and each member is able to buy one or more tickets. A particular type of ticket can be bought by multiple members of the club.

General & Specific Objectives of the Project:

General Objectives:

- The system must be laid out clearly
- The system must be easy to navigate around
- The system must use a similar design scheme as the website
- Members must have a unique membership number and password
- The system must cater for new members who want to join the club
- The system must be made simple for the club personnel to use
- The system must allow staff members to log in on an account
- The system must allow staff members to manipulate data
- The system must allow members to purchase club tickets with credit/debit
- Members must be able to contact the club via e-mail
- Members must be provided with a basic route to away grounds when selected

Specific Objectives:

- The system must provide boxes for people to enter their membership number and password to log on
- The home screen must have a button for new members to click and enter their details
- The member must be provided with an error message if they are unsuccessful logging on to the system with a reason why their number and password hasn't worked
- There must be a button for standard supporters to click if they want to gain membership and then they must be given somewhere to enter their basic details
- An error message must be returned to the supporter if their details are invalid
- The supporters details must be added to the data store if they are all valid
- Club personnel must be able to securely log on to the system and easily query club member's data when needed; this includes adding, deleting, updating and searching members details
- Members must be able to select the types and numbers of tickets they want to purchase
- There must be a secure method of payment for the member to use to buy tickets
- Members must be provided with a ticket confirmation after a successful purchase
- There must be an e-mail box on the system for members to write messages; these messages must then be able to be sent to an account created for the club
- An e-mail confirmation must be provided when the e-mail has been sent successfully
- Members must be able to select the name of the away team they are going to be travelling to
- A route must be displayed with basic directions to the chosen away ground

Potential Solutions and Justification of Chosen Solution:

Potential Solutions:

Solution Access:

- The system could be a second website running alongside the main Burgess Hill site. Whereas the official club site stores various news and information, the new website would be solely dedicated to the member features such as the club contact, ticket purchasing and route planner. Although, it would be logical to have separate sites for different purposes, this solution could cause confusion to people trying to access a specific site and could also prove time-consuming for the club personnel trying to manage and administrate two sites simultaneously. Further to this, another domain name for the new site would have to be registered with an internet registry
- The system could be integrated into the current club website where the source code of the site is updated with the new processes included. In terms of presentation, this is a worthy solution because all aspects of the site would have the same design scheme making it look professional to users. On the other hand, there is a possibility it would be difficult to navigate around because of the amount of pages contained on the site. Also, for people who didn't want to access any of the features of the new system, it would be more time-consuming to try and find the original information they required
- The system could be a separate program accessed from a section of the current site so the site remains unchanged but there is an easy way for members to get to the system. This would prove less complex for both staff and website users but may be difficult for members to find if it isn't linked in an obvious location on the website. A program could either be created as a console application which would be efficient to program and produce easily-readable code but would not provide a good graphical user interface unless a large proportion of time was spent making the graphics for each individual section. Alternatively, a program could be implemented with forms which would give a more professional look to the interface of the system but would require every form to be coded individually

Potential Implementation Languages:

- Implementing the system in Java is a possibility as it is compatible with many operating systems and the system can be connected to a database using JDBC which can be connected to Oracle and MySQL. However, I have little experience of the language and due to the time constraint, it may be difficult to start developing skills in Java as well as implementing the system
- If programming the system in C#, I could use the built-in Linq feature in order to connect to a member's database and do queries. But it could be hard to integrate the method of ticket purchasing because a server-side language would have to be used with C#
- The scripting language PHP is also an option, it is readily available on most public servers and there is good support for working with MySQL for the member's database. The language also integrates well with PayPal developer code. However, I have no prior knowledge of the language which could prove too ambitious with the time constraint

Potential File Types:

- CSV files could be used to store records of member's information as it would make querying details easier by simply searching files. However, with a growing number of members, this could be too time-consuming to search through a large file and there is a high chance of error occurring when manipulating multiple data in the file
- An SQL database would be more efficient in storing member's data because queries are able to be performed more easily and primary and foreign keys can be set for different tables of data. Additionally, the layout of the file is much clearer than a CSV file
- To read in values and text to the system for the route planner to calculate journeys, external CSV files could be used to store directions and distances
- The route planner could use simple external text files to read in data for the system to use
- A database could also be used to store information about routes but it could be unnecessary to create another table

Justification of Chosen Solution:

After assessing all the potential solutions, I have decided I am going to implement the majority of the solution in C# as it is the language I have the most experience with. I will be using multiple windows forms for each individual process which interlink with each other. They will also help to enhance the design of the system as well as making a user interface resembling the look of the club website. I will be able to create a server-based database and then use Linq queries for searching various records. Also, I can program all the necessary algorithms needed for the system in the language and read in any required files. I will use PayPal developer to simulate the ticket purchases because I can use a test account and show how tickets could be purchased in the system. Because of the nature of this solution, I will have to use HTML for a minority of the programming because PayPal developer requires server script code to function. The various values for the route planner will be read in from simple text files because I am not using records for the data. This solution will be accessed from the main website as the current members are familiar with the site and it will not have to interfere with the current content.

Design

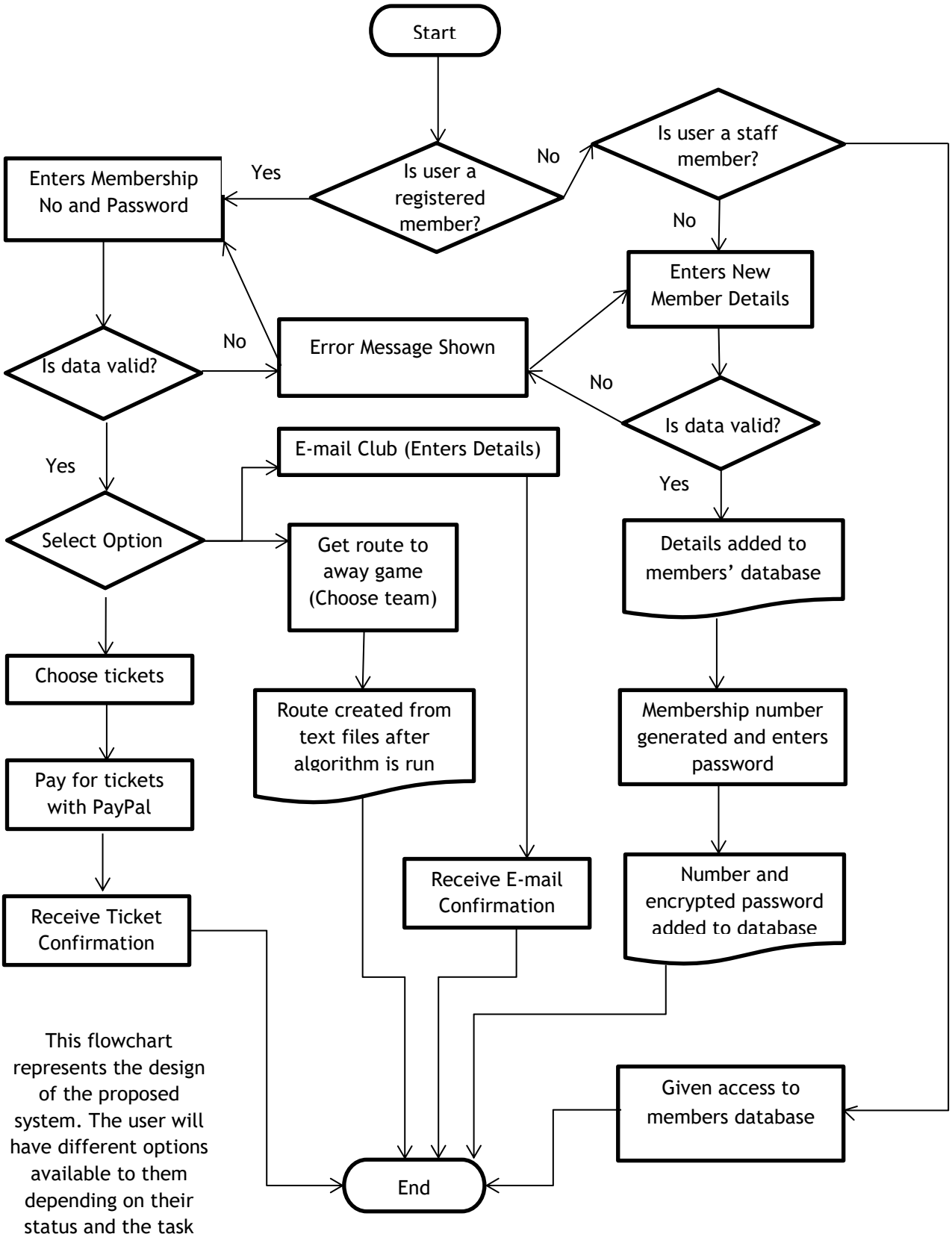
Overall System Design:

IPSO Chart:

Input	Processes	Storage	Output
• Membership Number • Password • Number of tickets • New Member Details • Away team • Option of journey in miles/kilometres	• Membership login • Calculating ticket costs • Purchasing tickets • Calculating away journey • Members to staff e-mailing • Generating membership number and storing passwords for new members • Encrypting passwords for security measures	• Members details in database • Login details for staff members in personnel table of database • Ticket details stored in database • Youth league team away directions in external file • Distance in miles external file • Distance in kilometres in external file • E-mail inboxes	• Ticket Confirmation • Account Confirmation • Route plan • New member confirmation • Results from searching members

This chart displays in a basic form the different ways in how the data in the new system is manipulated. It shows what is going to be inputted in and outputted from the system as well as the processes that will take place and proposes how the data will likely be stored.

System Flowchart:

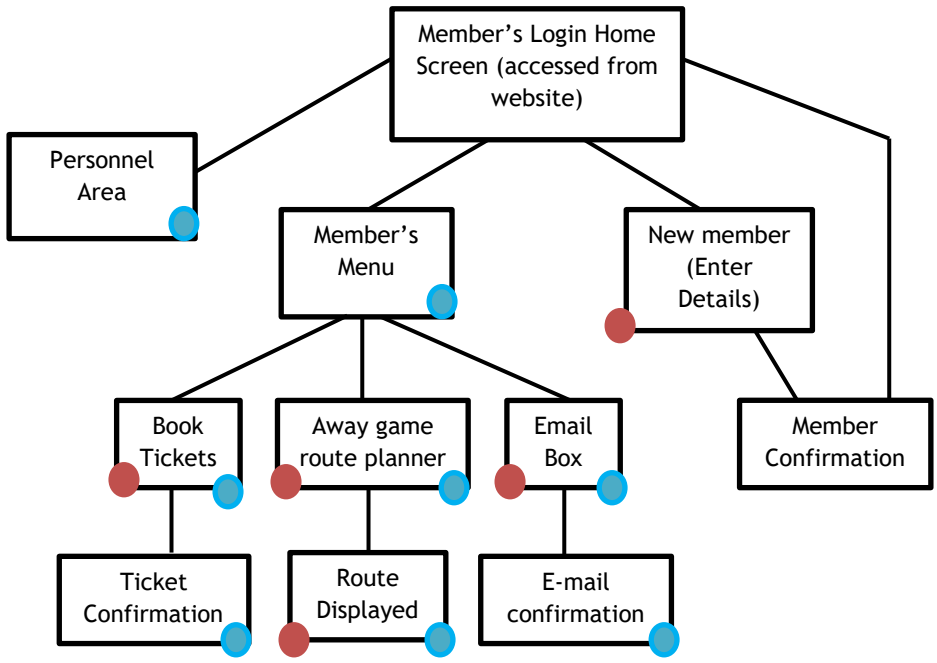


This flowchart represents the design of the proposed system. The user will have different options available to them depending on their status and the task they want to perform.

Modular Structure of System:

Form Navigation:

- Members home screen
  - Members Menu
    - Book tickets (PayPal)
      - Ticket Confirmation
    - Plan route for away game
      - Display route
    - Email queries
      - E-mail sent confirmation
  - New Member
    - Member Confirmation
  - Personnel Area



- The blue circle on the forms indicates that there will be a logout button on these forms and you will be able to log out of these forms and return to the home screen
- The red circle on the forms indicates that there will be a back button on these forms and you will be able to return to the previous page of the system

Design Data Dictionary:

Field	Data Type	Size/Range
Name	String	0 – 30
MembershipNo	Integer	8
PersonnelLoginNo	Integer	7
Password	String	8 - 20
CardType	String	
CardNumber	CreditCard	16
CardExpiryDate	Date	--/-- format
CardSecurityCode	Integer	3
Address	String	0 – 50
Postcode	String	7 LLNN NLL
MatchDate	DateTime	
TicketType	String	
TicketPrice	Currency	
TicketQuantity	Integer	1 - 5
TotalAmount	Currency	
EmailAddress	EmailAddress	0 – 40
PhoneNumber	Integer	11
EmailSubject	String	0 – 40 characters
EmailMessage	String	0 – 400 characters
AwayTeam	String	
AwayTeamGround	String	
AwayTeamAddress	String	
AwayTeamPostcode	String	
DistanceMiles	Single	
DistanceKilometres	Single	
Directions	String	

Validation Checks:

Field	Validation Check	Description	Valid Data	Erroneous Data	Error Message
Name	Data Type	Check name is entered as a string with no numeric data	John Smith	JohnSmith1	Re-enter name
	Range	Check name does not exceed 30 characters	John Smith	Joooohhhhhhnnn Sssmmiiiiiittthhh	Check name length
	Presence	Check not null	John Smith	null	Please enter a name
MembershipNo.	Data Type	Check no. only contains numeric data	12345678	AB345678	Ensure your membership number contains only numeric characters
	Length	8	12345678	1234567	Ensure your membership number is 8 characters
	Presence	Check not null	12345678	null	Please enter your membership number
	Data Presence	Check membership no. is present in members table	Number stored in database	Number not in database	There is no record of this membership number
PersonnelLoginNo	Data Type	Check no. only contains numeric data	1234567	AB34567	Ensure your login number contains only numeric characters
	Length	7	1234567	123456	Ensure your login number is seven characters
	Presence	Check not null	1234567	null	Please enter your login number
	Data Presence	Check login no. is present in personnel database table	Number stored in database	Number not in database	There is no record of this login number
Password	Data Type	Check password contains both letters and numbers	Password12	Password	Your password needs to contains letters and numbers

Field	Validation Check	Description	Valid Data	Erroneous Data	Error Message
	Range	Check password is 8 characters or more and doesn't exceed 20 characters	Password12	Pass12	Password needs to be between 8 and 20 characters long
	Presence	Check not null	Password12	null	Please enter your password
CardType	Lookup	Select from drop down menu	Visa		
	Presence	Check card type has been selected	Visa	null	Please select a card type
CardNumber	DataType	Check number contains all integers	1111 1111 1111 1111	A111 1111 1111 1111	Check card number
	Length	Check number has 16 digits	1111 1111 1111 1111	1111 1111 1111 1111 1111	Check card number
	Presence	Check not null	1111 1111 1111 1111	null	Please enter your card number
CardExpiryDate	Lookup	Check date is after current date	02/17		
	Presence	Check date has been selected	02/17	null	Please select your card expiry date
CardSecurityCode	DataType	Check number is all integers	475	A75	Ensure your code is a number
	Length	Check the number is three digits	475	4750	Check code is 3 digits
Address	Presence	Check not null	20 Example Lane, Example Road, Example Town	null	Please enter your address
Postcode	DataType	Check postcode is a string in LLNN NLL L = letter N = number	RH15 1AZ	RH15 ZZZ	Check your postcode is correct
	Length	Check postcode is 4 characters followed by a space followed by 3 characters	RH15 1AZ	RH115 11AZ	Ensure your postcode is the right number

Field	Validation Check	Description	Valid Data	Erroneous Data	Error Message
	Presence	Check not null	RH15 1AZ	null	Please enter your postcode
TicketType	Lookup	Select from drop down menu of ticket types	Member		
	Presence	Check not null	Member	Not selected	Ensure you have selected a ticket type
TicketQuantity	Lookup	Select from drop down menu of ticket quantities	2		
	Presence	Check not null	2	Not selected	Ensure you have selected a ticket quantity
EmailAddress	Presence	Check not null	hannah.short1@-btinternet.com	null	Ensure you have entered an e-mail address
Phone Number	DataType	Check phone number contains all integers	01444 123456	ABCDE 123456	Ensure you have entered a correct phone number
	Length	Check number is 11 digits long	01444 123456	01444 12345678	Ensure your number is only 11 digits
	Presence	Check not null	01444 123456	null	Ensure you have entered a phone number
EmailSubject	Range	Check subject doesn't exceed 40 characters	*Subject with 40 or less characters*	*Subject with more than 40 characters*	Ensure your subject is less than 40 characters
	Presence	Check not null	*subject*	null	Ensure you have entered a subject for the e-mail
EmailMessage	Range	Check message is not over 400 characters	*subject with 400 characters or less*	*subject with more than 400 characters*	Ensure your message is less than 400 characters
	Presence	Check not null	*message*	null	Ensure you have entered your e-mail message
AwayTeam	Lookup	Select team from drop down menu	Bognor Regis Football Club		
	Presence	Check team has been selected	Bognor Regis Football Club	null	Ensure you have selected an away team

Record Structure Description:

The system will be reading in data when producing the route to the away ground. This data will be read in from .txt files because the data will not be in records or arrays. I will have separate text files, one containing all the possible qualitative directions to all the clubs, one with all the weights from the graph in miles and one with all the weights in kilometres. The file with the distances which is read from will depend on whether the user selects to view their journey in miles or kilometres.

Data for text files:

Away Team	Distance from club (miles)	Distance from club (km)	Journey Time (minutes)
Bognor Regis	36.7	59.1	62
Eastbourne Borough	32.3	52	51
Eastbourne Town	31	49.9	56
Hastings United	39.6	63.7	80
Horsham	14.8	23.8	28
Lewes	12.4	20	29
Three Bridges	15	24.1	26
Whitehawk	16.5	26.6	36
Worthing	20.1	32.3	37

Away Team	Distances between successive nodes (miles)
Bognor Regis	3.0, 5.1, 3.3, 6.7, 12.6, 6.1
Eastbourne Borough	0.8, 6.9, 3.8, 6.4, 8.7, 5.7
Eastbourne Town	0.8, 6.9, 3.8, 6.4, 8.7, 4.4
Hastings United	0.8, 6.9, 3.8, 6.4, 8.7, 13.0
Horsham	3.0, 5.1, 6.7
Lewes	4.7, 4.8, 2.9
Three Bridges	4.3, 7.6, 3.1
Whitehawk	0.8, 6.9, 3.8, 2.1, 2.9
Worthing	0.8, 6.9, 11.2, 1.8

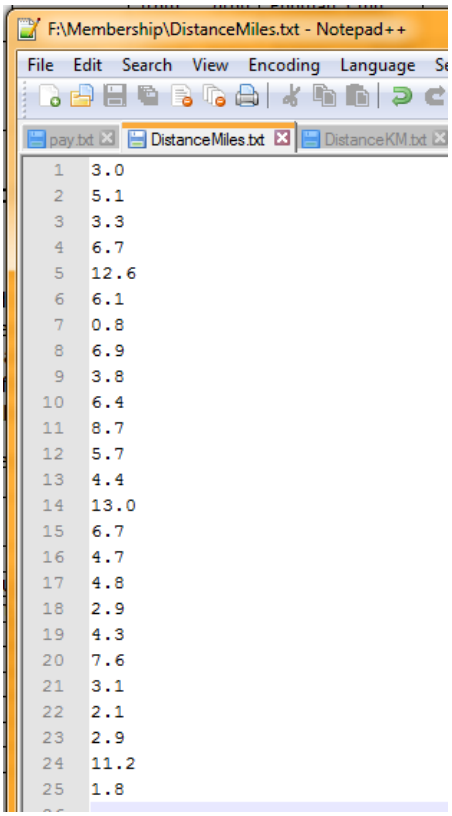
Away Team	Distances between successive nodes (km)
Bognor Regis	4.8, 8.2, 5.3, 10.8, 20.3, 9.8
Eastbourne Borough	1.3, 11.1, 6.1, 10.3, 14.0, 9.2
Eastbourne Town	1.3, 11.1, 6.1, 10.3, 14.0, 7.1
Hastings United	1.3, 11.1, 6.1, 10.3, 14.0, 20.9
Horsham	4.8, 8.2, 10.8
Lewes	7.6, 7.7, 4.7
Three Bridges	6.9, 12.2, 5.0
Whitehawk	1.3, 11.1, 6.1, 3.4, 4.7
Worthing	1.3, 11.1, 18.0, 2.9

Away Team	Directions
Bognor Regis	• Exit Burgess Hill by roundabout at bottom of London road taking 1st exit, Cuckfield Road and head towards Ansty
	• At roundabout, take 1st exit on Bolney Road (A272) • Carry on to Cowfold Road
	• Stay on A272 until A24 is reached
	• Turn right onto A24 and carry on until Washington
	• At roundabout, take 3rd exit to Storrington Road • At roundabout, take 1st exit onto High Street • At roundabout, take 2nd exit onto Amberley Road • At roundabout, take 2nd exit onto A29 • At roundabout, take 2nd exit onto A27
	• At roundabout, take 2nd exit onto Fontwell Avenue • At roundabout, take 2nd exit onto Nyton Road • At roundabout, take 1st exit onto A29 • Continue on A29 and then at roundabout, take 4th exit onto Chichester Road • Turn left onto Hawthorn Road • Turn left onto Nyewood Lane • Destination is on right
Eastbourne Borough	• Exit Burgess Hill via roundabout at top of London Road
	• Follow A273 through Keymer and Clayton
	• At roundabout, take the 2nd exit onto A27
	• Stay on A27
	• At Beddingham roundabout, take 1st exit and stay on A27
	• At roundabout, take 2nd exit onto A22 • At roundabout, take 1st exit onto Willingdon Drove • Continue straight onto B2104 • At roundabout, take 3rd exit onto Priory road • Turn left onto Priory Lane • Destination is on right

Away Team	Directions
Eastbourne Town	• Exit Burgess Hill via roundabout at top of London Road
	• Follow A273 through Keymer and Clayton
	• At roundabout, take the 2nd exit onto A27
	• Stay on A27
	• At Beddingham roundabout, take 1st exit and stay on A27
	• At Willingdon roundabout, take the 2nd exit onto Willingdon Road • Continue onto Upperton Road • At roundabout, take 2nd exit onto Grove Road • Turn right onto Meads Lane • Destination is on right
Hastings United	• Exit Burgess Hill via roundabout at top of London Road
	• Follow A273 through Keymer and Clayton
	• At roundabout, take the 2nd exit onto A27
	• Stay on A27
	• At Beddingham roundabout, take 1st exit and stay on A27
	• At roundabout, take 2nd exit onto A259 • Turn left onto Bexhill Road • At the roundabout, take the 2nd exit onto B2204 • Turn right onto Telham Lane • Turn right on Hastings Road • At roundabout, take 1st exit onto The Ridge • Turn right onto Elphinstone Road • Destination will be on left
Horsham	• Exit Burgess Hill by roundabout at bottom of London road taking 1st exit, Cuckfield Road and head towards Ansty
	• At roundabout, take 1st exit on Bolney Road (A272) • Carry on to Cowfold Road
	• At roundabout, take 2nd exit onto the A281 and follow straight on to Horsham • Turn left onto Queensway • Turn left onto The Hornets • Destination is on left

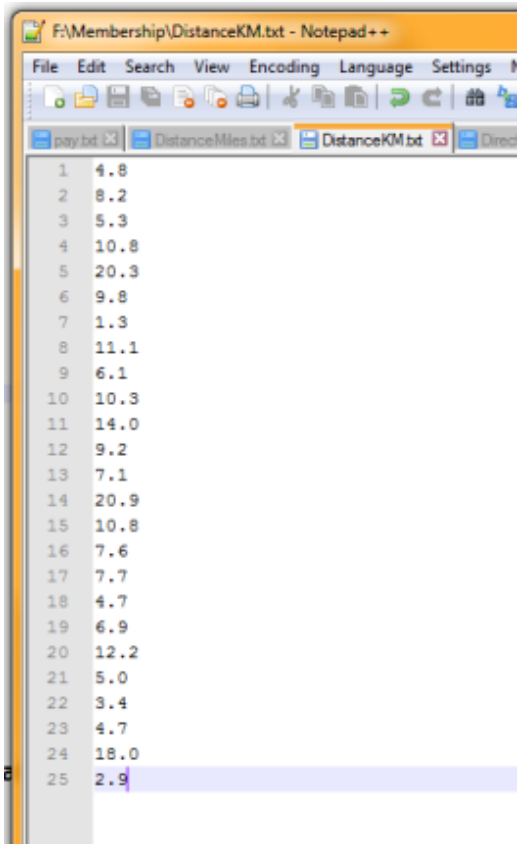
Away Team	Directions
Lewes	• Head out the top of Burgess Hill on Janes Lane and turn right on the Ditchling Road • Take 1st exit at roundabout onto Folders Lane East • Keep right onto Spatham Lane
	• At Ditchling, turn left onto Lewes Road • Continue to follow B2116
	• Turn right onto A275 • Keep left on A2029 • Turn left onto Station Street • At roundabout, take 1st exit and continue to Mountfield Road • Destination is on right
Three Bridges	• Drive out of Burgess Hill via the A2300 and onto the A23
	• Continue on A23 towards Crawley
	• At roundabout, take 3rd exit onto Southgate Avenue • Turn right onto Hawthe Avenue • At roundabout, take 3rd exit onto Haslett Avenue • Turn left onto Three Bridges Road • Turn left onto Jubilee Walk • Destination is on right
Whitehawk	• Exit Burgess Hill via roundabout at top of London Road
	• Follow A273 through Keymer and Clayton
	• At roundabout, take the 2nd exit onto A27
	• At roundabout, take 3rd exit onto B2123
	• Turn right onto Warren Road • Turn left onto Wilson Avenue • Destination is on right
Worthing	• Exit Burgess Hill via roundabout at top of London Road
	• Follow A273 through Keymer and Clayton
	• At the roundabout, take 1st exit onto A27 • At roundabout, take 2nd exit onto Upper Brighton Road • At the roundabout, take the 1st exit onto A24
	• Turn right onto A2032 • At the roundabout, take 3rd exit onto Bulkington Avenue • Turn left onto Woodside avenue • Destination straight ahead

Distance in miles text file:



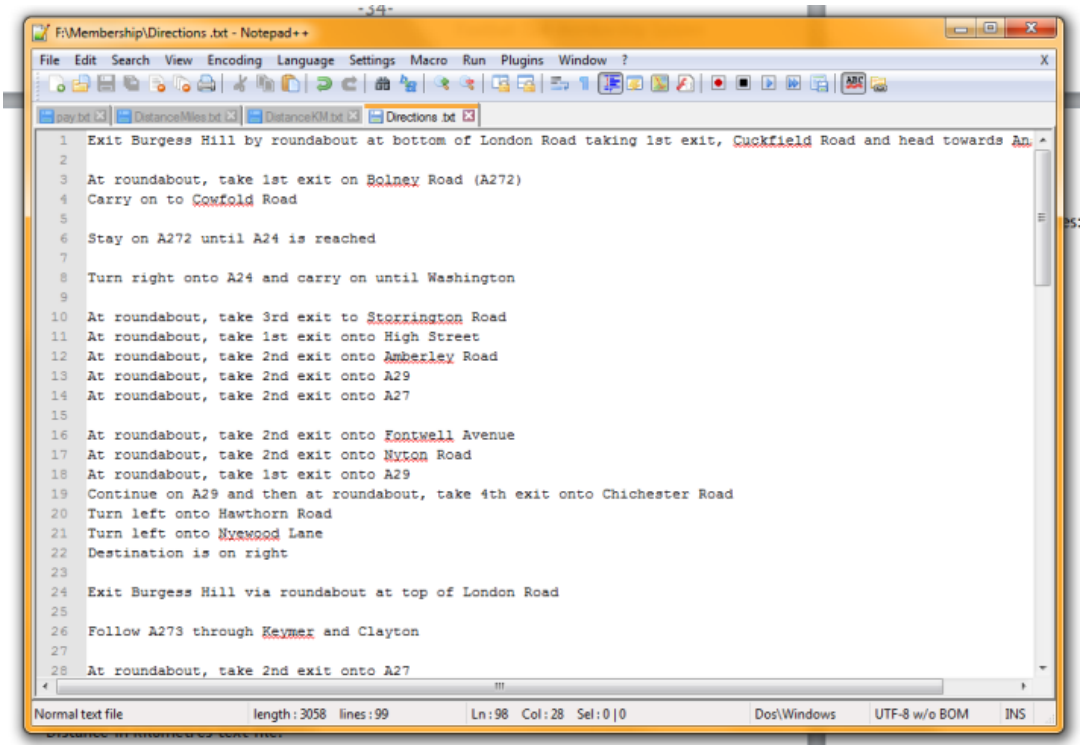
This is the text file that will store all the weights of the nodes on the graph in miles if the user chooses to have their journey displayed in miles

Distance in kilometres text file:



This is the text file that will store all the weights of the node on the graph in kilometres if the user chooses to have their journey displayed in kilometres

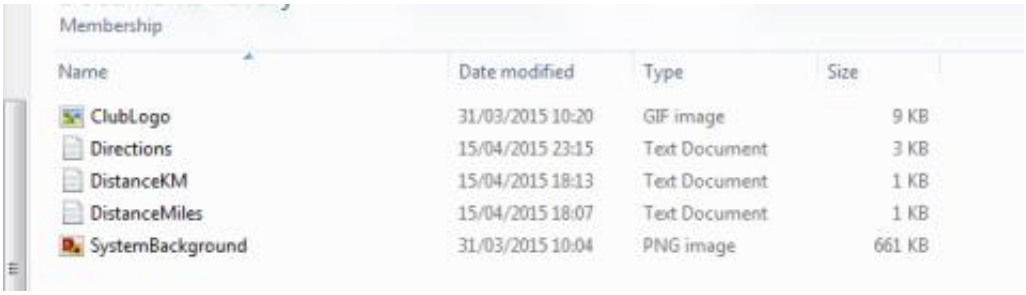
Directions text files:



This is the text file that will store all the directions to each club, each set of directions corresponds to one of the weightings on the graph so it is easier to match a length with the corresponding directions when the text files are being read from.

File Organisation and Processing:

All the relevant files to the system will be kept in a folder. These will include the text files, the project file with the project folder, images for the design and the database.

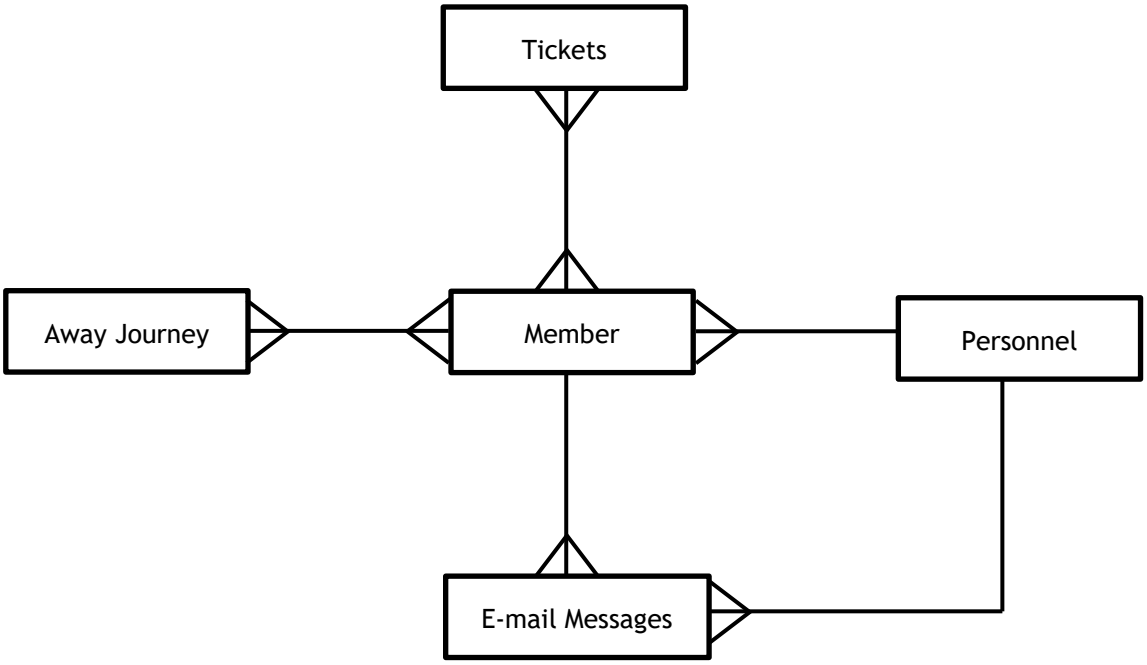


The text files have very little memory and the club logo that I will be using on the forms is relatively small. I have designed a background for the forms which is larger due to the editing. I predict that the project itself will take up little space whereas the files in the project folder will take up a lot. Initially, the sql file for the database will be reasonably small but when it is populated with test data, it should significantly increase in size.

Database Design:

Entity-Relationship Diagram for Proposed System:

The new system will be using a database to store and keep track of the details of the club members.

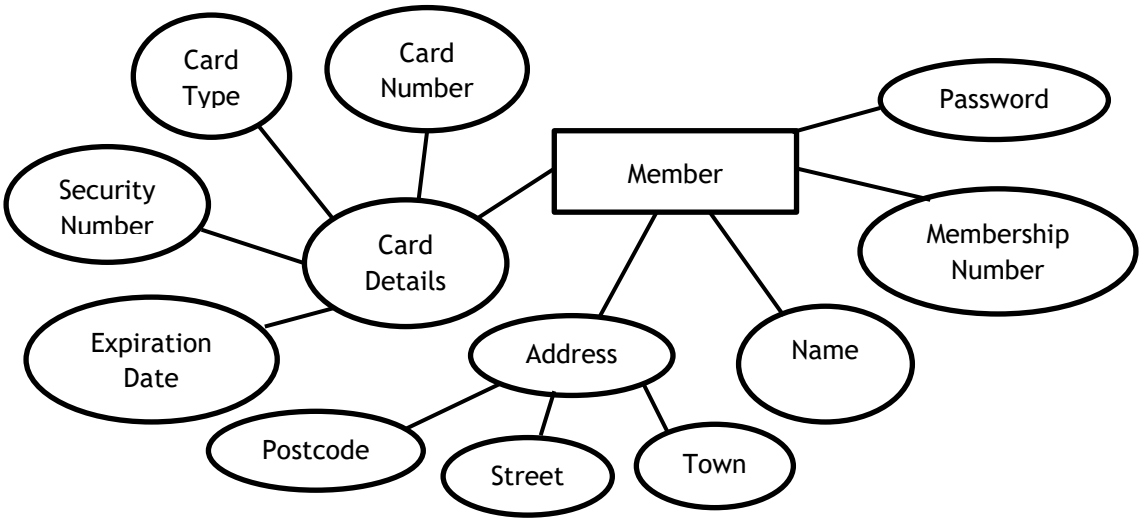


This diagram shows the relationships between the entities in the system. A member can purchase multiple tickets and the same type of ticket can be bought by several members. The staff at the club will be responsible for the data and purchases made by multiple members and they will be able receive e-mails from any members. Lastly, a member can request routes to different away grounds and a specific route may be requested by more than one member.

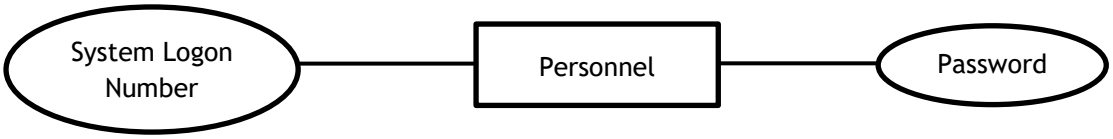
Entity Attributes:

These diagrams show the potential attributes of each entity involved in the system. These are the properties and characteristics of the specified object and are column headings for a potential database solution.

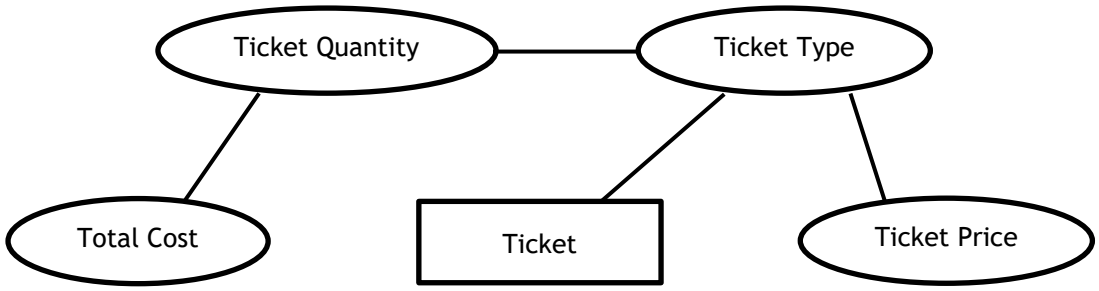
Attributes of Member Entity:



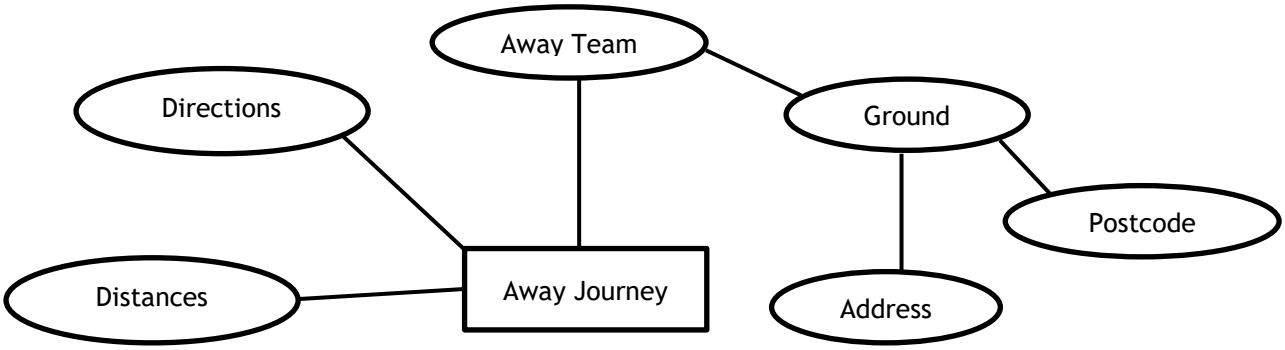
Attributes of Personnel Entity:



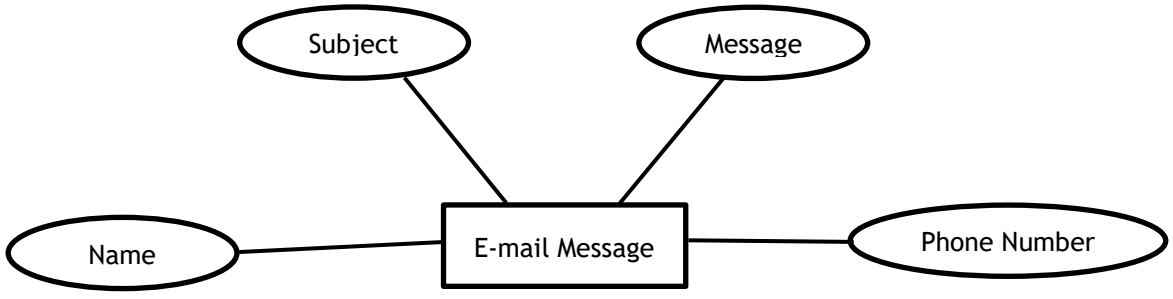
Attributes of Ticket Entity:



Attributes of Away Journey:



Attributes of E-mail Message:



Normalised Relations:

The tables in my database will have to be fully normalised where all attributes are dependent on the key and there are no repeating groups of attributes to aid the consistency of the data. In the member table, the primary key will be membership number whereas in the personnel table it will be login number.

Member(MembershipNo, Password, FName, SName, Address, Postcode)

Personnel(LoginNo, Password)

Tickets(MembershipNo, TicketType, TicketCost, TicketQuantity, TotalCost)

Design of Tables:

Members Table:

Membership Number	Password	FName	SName	Address	Postcode

Personnel Table:

Login No	Password

Tickets Table:

Membership Number	TicketType	TicketCost	TicketQuantity	TotalCost

Planned SQL Queries:

Club personnel will need to be able to query members' data to check for particular details. Also, when the member enters their membership number and password in the login screen, the member's database will need to be queried to make sure the data exists in the relevant table. Here are all the possible queries that will be able to be performed written in structured query language:

- Searching and Displaying a particular member

  This query will be performed when a club member or member of staff clicks the logon button or a member of staff clicks the search member button on the personnel control form

  Example of the query when a member attempts to log on to the system:

```
SELECT  MembershipNo
FROM Members
WHERE MembershipNo = MembershipNoTextBox.Text
AND (SELECT Password
        FROM Members
        WHERE Password = PasswordTextBox.Text)
```

  Example of the query when a member of staff attempts to log on to the system:

```
SELECT  LoginNo
FROM Personnel
WHERE LoginNo = MembershipNoTextBox.Text
AND (SELECT Password
        FROM Personnel
        WHERE Password = PasswordTextBox.Text)
```

Example of query when a members details are searched for on the personnel control form:

```
SELECT  MembershipNo
FROM Members
WHERE MembershipNo = MembershipNoTextBox.Text

SELECT  Password
FROM Members
WHERE Password = PasswordTextBox.Text

SELECT  FName
FROM Members
WHERE FName = FNameTextBox.Text

SELECT  Surname
FROM Members
WHERE Surname = SurnameTextBox.Text

SELECT  Address
FROM Members
WHERE Address = AddressTextBox.Text

SELECT  Postcode
FROM Members
WHERE Postcode = PostcodeTextBox.Text
```

When one of the conditions are met, all the other variable will be displayed in the respective boxes

For example, if the postcode entered by the staff member:

```
SELECT  Postcode
FROM Members
WHERE Postcode = Postcode in Postcode Text Box

If Postcode valid (PRINT MembershipNo, Password, FName,
Surname, Address)
```

- Add a new member to database
  In the personnel control form and the new members form, a new member will be able to be added to the database
  In the new member form the user will enter their first name and surname with their address and postcode and if these are valid they will be added to the database. Later on, a membership number and password will be created and these will be added to the same record
  In the personnel control form, all the data including the membership number and password can be entered at once and added to the members table

```
INSERT INTO Members (MembershipNo, Password, FName, Surname, Address, Postcode)
MEMBER (MembershipNoTextBox.Text, PasswordTextBox.Text, FNameTextBox.Text, SurnameTextBox.Text, AddressTextBox.Text, PostcodeTextBox.Text)
```

- Editing an existing members details
  In the personnel form, the data of a member can be edited. When the button is clicked, the query will match the membership number that the member of staff has entered in the membership number textbox with a corresponding one in the database and then overwrite the rest of the data fields with the data in the other text boxes

```
UPDATE Member
SET Password = PasswordTextBox.Text, FName = FNameTextBox.Text, Surname = SurnameTextBox.Text, Address = AddressTextBox.Text, Postcode = PostcodeTextBox.Text
WHERE MembershipNo = MembershipNoTextBox.Text
```

- Delete a member

```
DELETE FROM Members
WHERE MembershipNo = MembershipNoTextBox.Text
```

Data Definition Language:

The creation of the tables in the database will be defined by data definition language

DDL for Members Table:

```
CREATE TABLE Members
   MembershipNo INT PRIMARY KEY (NOT NULL)
   Password VARCHAR (15)
   FName VARCHAR (15)
   SName VARCHAR (15)
   Address VARCHAR (50)
   Postcode VARCHAR (15)
```

DDL for Personnel Table:

```
CREATE TABLE Personnel
   LoginNo INT PRIMARY KEY (NOT NULL)
   Password VARCHAR (15)
```

DDL for Tickets Table:

```
CREATE TABLE Tickets
   MembershipNo INT PRIMARY KEY (NOT NULL)
   TicketType VARCHAR (15)
   TicketCost CURRENCY
   TicketQuantity INT
   TotalCost CURRENCY
```

Storage Media Identification:

The membership system program will need to be stored on a specific type of storage media to give to the football club to integrate into their website. The media will need to be able to store all the relevant files for the system including the program, the database, the text files and any graphics.

I am going to evaluate several types of media in order to decide which type will be most suitable to store the system on.

Magnetic Hard Disk

I could be stored on a magnetic hard disk as they are suitable for storing programs and data files on. However, their capacities range up to terabytes which would be unnecessary for this purpose because that amount of memory would never be used and would therefore be wasted. Although hard disks have incredibly fast transfer speeds and access time, it wouldn't be appropriate to use one in this situation.

Magnetic Floppy Disk

A magnetic floppy disk is useful for backing up and transferring small files but these usually tend to be small as there is a very limited amount of storage on them. There is also the issue that floppy disk drives are rare in computers nowadays so it would be likely the club would have no way of accessing the content on the disk if their desktop computer didn't have a floppy disk drive.

CD-ROM

A CD-ROM is able to store a sufficient amount of memory for this purpose and the club is much more likely to have a CD/DVD drive than they are a floppy disk drive. Furthermore, it is easier to distribute software on a CD than it is most other types of media and the initial files on the disk will not be able to be overwritten before they are copied onto the user's system.

USB Flash Drive

A USB stick offers a generous storage capacity and it is highly likely the computer the system is run on will have several USB ports. There is also the benefit that a USB flash drive is small, lightweight and robust, but its size can also mean that it is easy to misplace. However, they are extremely compact, have a fast transfer speed and can hold a lot of data.

Overall, the system would be most suitably stored on either a CD-ROM or a USB flash drive. Both mediums have appropriate storage capacities for the files and are suitable for distributing files. However, I will be choosing to store the system on a USB flash drive because they are more commonly used than CDs today and are more reliable due to their design. Also, there is the option to install the system on alternative devices if needed such as laptops because USB ports are present on these.

Algorithm Identification:

Password Encryption:

When a new member is added to the system, the password they choose will need to be encrypted when it is stored in the members' database. I am going to use a substitution cipher. Although, a cipher is relatively easy to break, security at the club does not seem to be a great issue and there are very few members of staff who will be using the system.

Substitution cipher:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D

Pseudo-Code Algorithm:

```
EncryptPassword
 count ← 0
 do until count = password length
   count = count + 1
   get character at position count
   letter[count] = ASCII(letter[count])
   letter[count] = letter[count] + 4
   if 122 < letter[count] < 127 then letter[count] = 97 + (letter[count] –
123)
 loop
 count ← 0
 do until count = password length
   count = count + 1
   (letter[count]) = CHR(letter[count])
 loop
end EncryptPassword
```

The algorithm will shift each letter four places back in the alphabet by adding four to the ASCII value of each letter in the password.

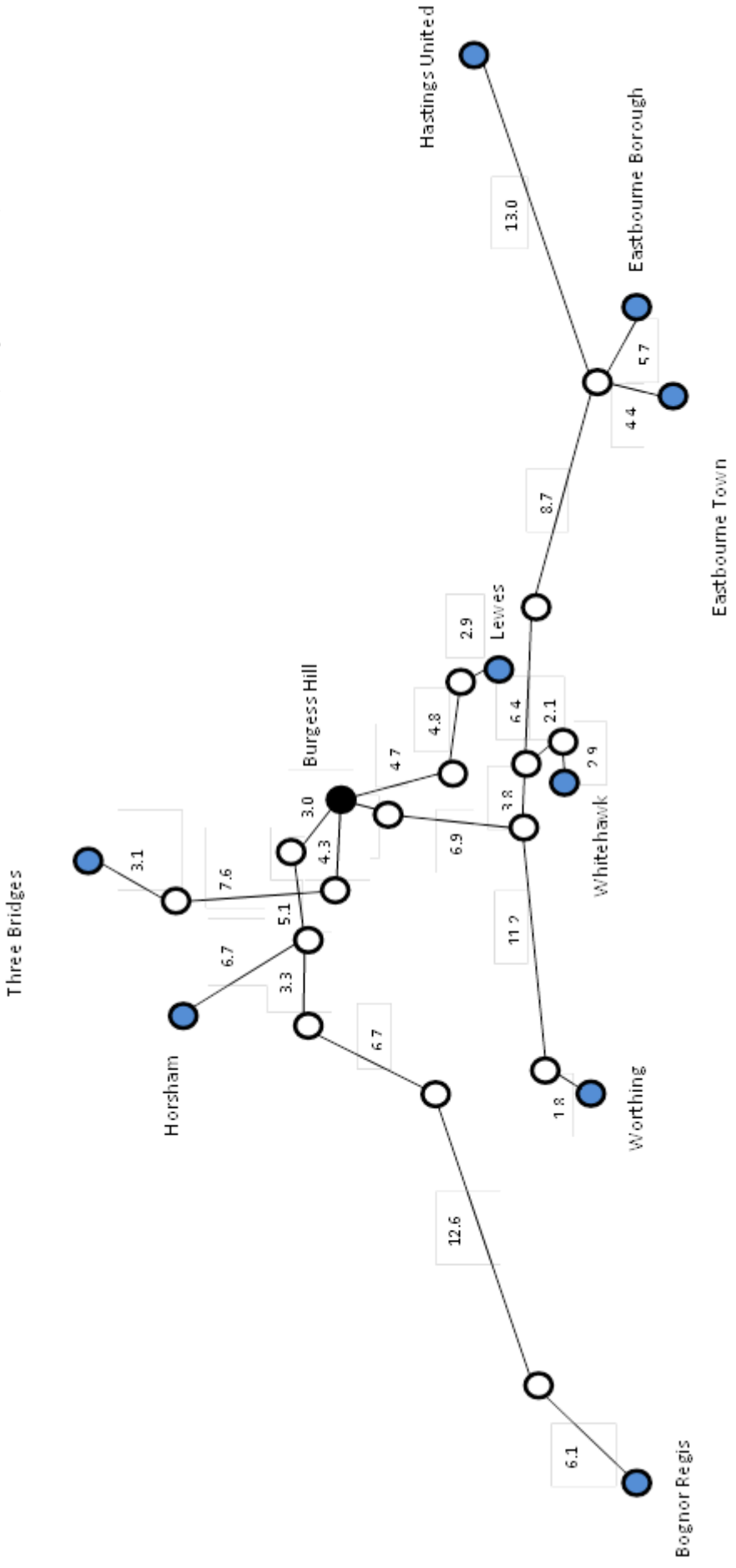
Route Planner Algorithm:

To find the correct route from Burgess Hill football club's ground to the destination of the away team's ground, I will use a tree traversal algorithm.

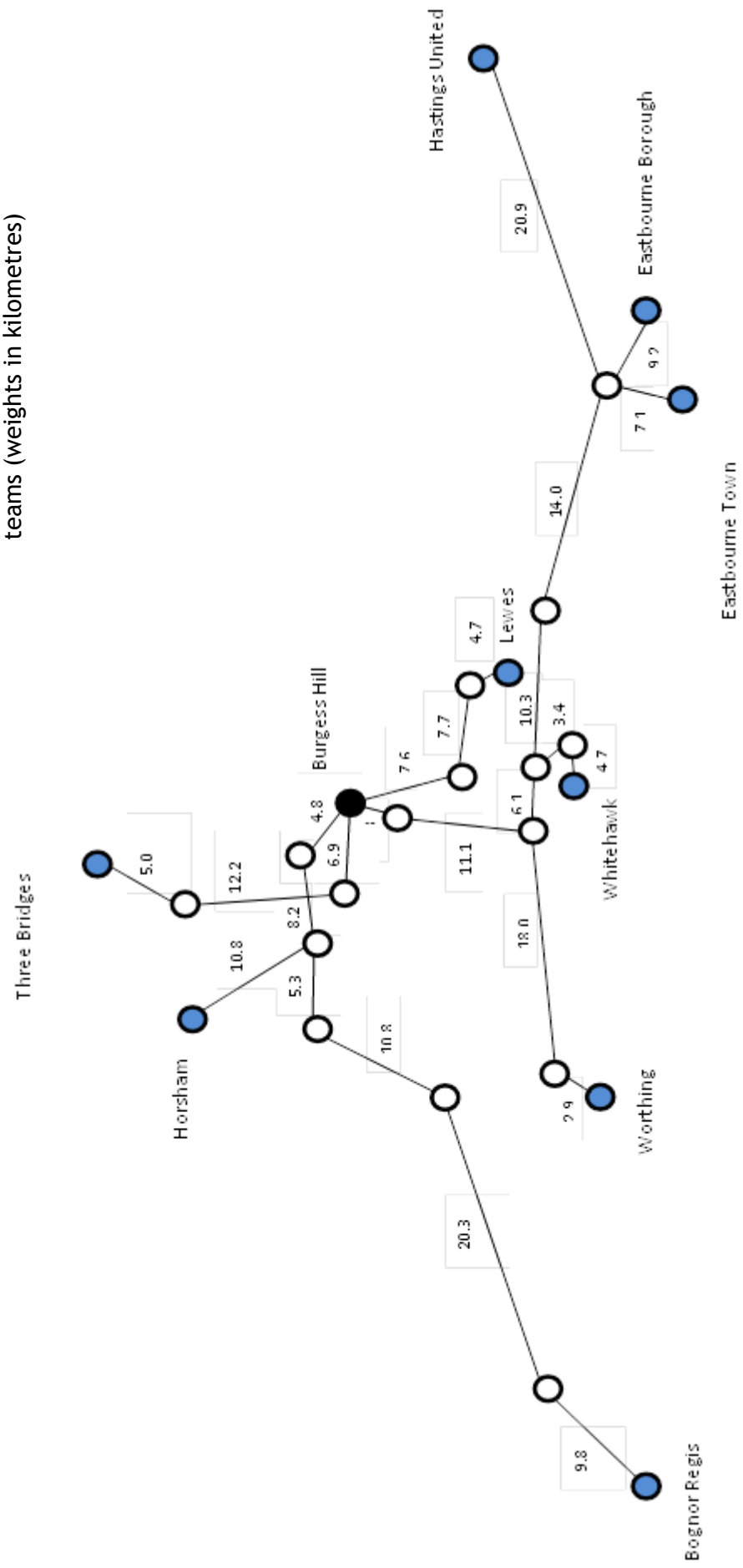
Pseudo-Code Algorithm:

```
GetRoute (Node, EndNode)
Found[Node] ← true
 if Node = EndNode then Reached ← true
 for each NextNode N linked to Node
  if Found[N] = false then GetRoute(Node,EndNode)
 end for
 DestinationReached[Node] ← true
 Output found nodes
End GetRoute
```

Graph of the grounds of the away teams (weights in miles)



Graph of the grounds of the away teams (weights in kilometres)



Entry Validation Algorithms:

There are a number of attributes involved in the system that will need to be validated when they are entered by the user. The system needs to check data types, sizes and that there is not null data.

- Membership Number Validation

  When a user logs on to the membership system, the membership number they enter will need to be check to ensure that it has the right number of digits, contains only numerical data and that they have not left the text box empty. Also after these checks, the number will need to be searched for in the members' database to make sure it exists.

  Pseudo-Code Algorithm:

```
if MembershipNo = null
  PRINT "Please enter your membership number"
else
if Length(MembershipNo) <> 8
  PRINT "Ensure your membership number is 8 digits"
else
forEach digit in number
  If digit <> numeric
    PRINT "Ensure your membership number contains only numeric
characters"
else
  SELECT MembershipNo
  FROM Members
       WHERE MembershipNo = MembershipNoTextBox.Text
  if forEach
        Members.MembershipNo <> MembershipNoTextBox.Text
          PRINT "There is no record of this membership number"
end
```

- Personnel Login Number Validation

  Likewise, when a member of staff needs to access the system, the login number they have entered will need to be validated before the database containing personnel login details is searched to ensure the number exists.

  Pseudo-Code Algorithm:

```
if LoginNo = null
  PRINT "Please enter your login number"
else
```

```
if Length(MembershipNo) <> 7
  PRINT “Ensure your login number is 7 digits”
else
forEach digit in number
  If digit <> numeric
    PRINT “Ensure your login number contains only numeric
characters”
else
  SELECT LoginNo
  FROM Personnel
       WHERE LoginNo = MembershipNoTextBox.Text
  if forEach
        Personnel.LoginNo <> MembershipNoTextBox.Text
          PRINT “There is no record of this login number”
end
```

- Password Validation

  When a member or staff member logs in, the system needs to ensure that they have entered a password into the password text box, the password is an appropriate length and that the password entered belongs to the same user as the membership number entered

  Pseudo-Code Algorithm:

```
if Password = null
  PRINT “Please enter your password”
else
if length(Password) < 8 or length(Password) > 20
  PRINT “Your password needs to between 8 and 20 characters”
else
if
  SELECT Password
  FROM Members
    WHERE Members.MembershipNo = MemberhipNoTextBox.Text
    AND Members.Password = PasswordTextBox.Text
== false
  PRINT “Invalid Password”
end
```

- New Member Validation

  On the new member form, all the data that the user enters will have to be validated by the system. The first name and surname will have to be checked to make sure they only contain alpha characters and the postcode for a correct format.

Pseudo-Code Algorithms:

Validate First Name:

```
if FName = null
  PRINT “Please enter your first name”
else
forEach character
  if character = numeric
    PRINT “Please ensure your first name contains no numeric
characters”
else
  if length(FName) > 30
    PRINT “Please ensure your first name is under 30 characters”
end
```

Validate Surname:

```
if Surname = null
  PRINT “Please enter your first name”
else
forEach character
  if character = numeric
    PRINT “Please ensure your first name contains no numeric
characters”
else
  if length(Surname) > 30
    PRINT “Please ensure your first name is under 30 characters”
end
```

Validate Address:

```
If address = null
 PRINT “Please enter your address”
end
```

Validate Postcode:

```
if Postcode = null
  PRINT “Please enter your postcode”
else
if length(Postcode) > 7
  PRINT “Please ensure your postcode contains 7 characters”
else
 if 1st, 2nd, 6th and 7th characters = numeric
  or
  3rd, 4th, 5th characters <> numeric
```

```
  PRINT “Please ensure your postcode is valid”
end
```

- E-mail Validation

  On the e-mail form, when the member enters their details into the text boxes they will need to be validated to reduce the chances of the club being sent erroneous data. All fields will have to be check to make sure they have been entered; the system will need to ensure the name boxes contain no numeric characters whilst the phone number box contains only numeric characters.

  Validate First Name:

```
if FName = null
  PRINT “Please enter your first name”
else
forEach character
  if character = numeric
    PRINT “Please ensure your first name contains no numeric characters”
else
  if length(FName) > 30
    PRINT “Please ensure your first name is under 30 characters”
end
```

  Validate Surname:

```
if Surname = null
  PRINT “Please enter your first name”
else
forEach character
  if character = numeric
    PRINT “Please ensure your first name contains no numeric characters”
else
  if length(Surname) > 30
    PRINT “Please ensure your first name is under 30 characters”
end
```

  Validate Phone Number:

```
if PhoneNumber = null
  PRINT “Please enter your phone number”
else
forEach digit
   if digit <> numeric
```

```
     PRINT "Please ensure your phone number contains only numeric
characters"
else
if length(PhoneNumber) <> 11
    PRINT "Please ensure your phone number is 11 digits"
end
```

Validate E-mail Address:

```
if emailAddress – null
  PRINT "Please enter your e-mail address"
end
```

Validate E-mail Subject:

```
if emailSubject – null
  PRINT "Please enter a subject for your e-mail message"
else
if length(emailSubject) > 40
  PRINT "Please ensure your subject doesn't exceed 40 characters"
end
```

Validate E-mail Message:

```
if emailMessage – null
  PRINT "Please enter a subject for your e-mail message"
else
if length(emailMessage) > 400
  PRINT "Please ensure your subject doesn't exceed 400 characters"
end
```

Ticket Price Calculation:

When the member has chosen their tickets types and quantities, the system will be required to perform a simple calculation of the total cost of all the tickets.

```
CalcTotalCost(AdultTicketQuantity, MemberTicketQuantity)
 AdultsCost = AdultTicketQuantity * 9
 MembersCost = MemberTicketQuantity * 5
 TotalCost = AdultsCost + MembersCost
endCalcTotalCost
```

The algorithm will only need to take the pricing of adult and members tickets into account because when an under 18 ticket is selected on the ticket form, there is no charge.

Membership Number Generation:

When a user enters their details into the new members form, if they are valid a membership number will be generated for them and displayed on the next form. The algorithm will be required to generate the lowest valid membership number which doesn't already exist in the members' database.

Pseudo-Code Algorithm:

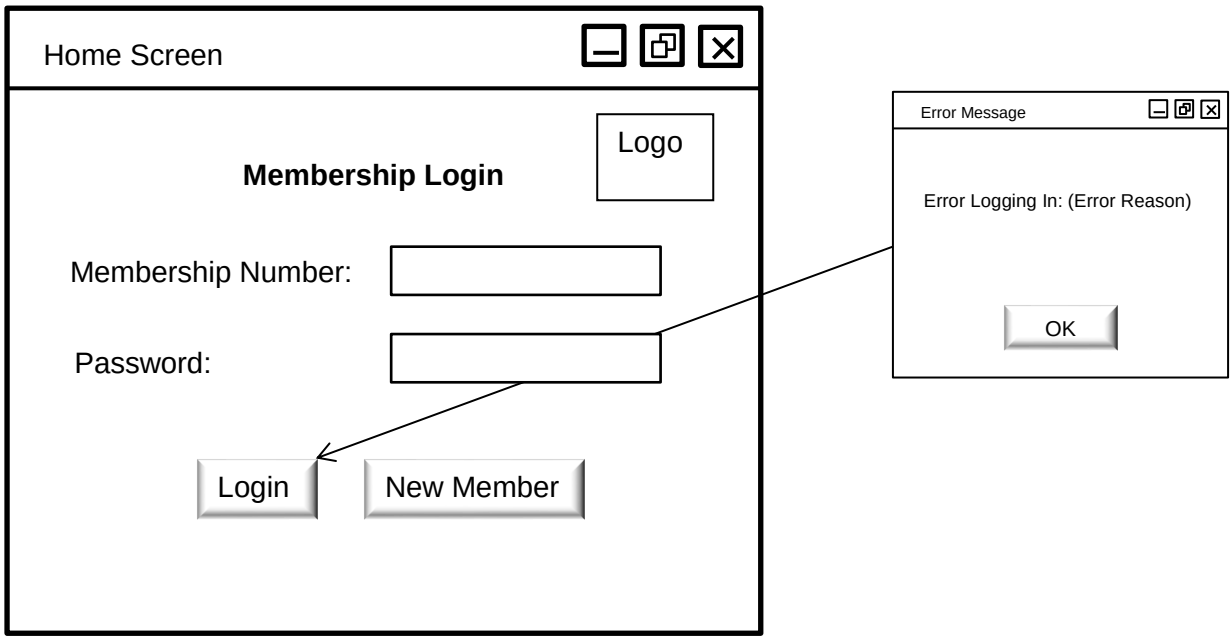
```
GenerateNumber
 NewMembNumber = Max(Members.MembershipNo) + 1
endGenerateNumber
```

User Interface Design:

Planned Data Capture & Entry Designs and Planned Valid Output Designs:

Home Screen Form:

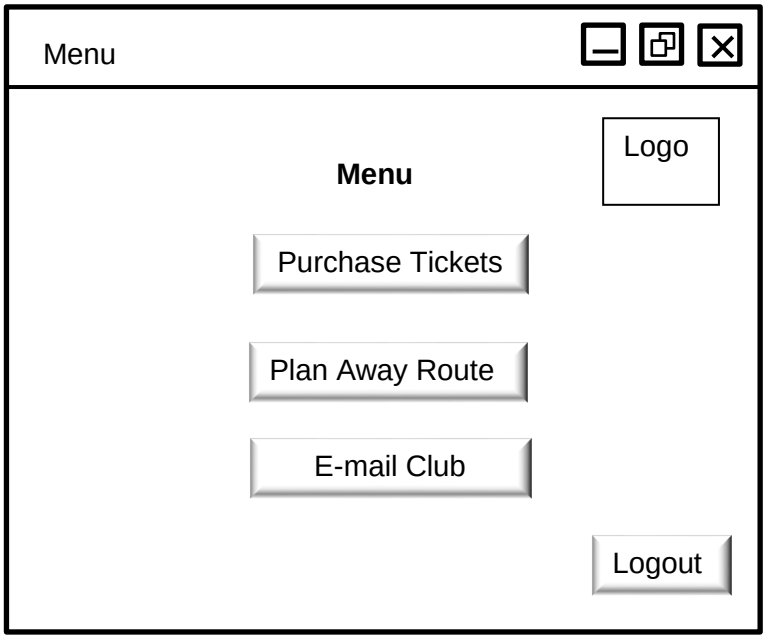
The home screen will be the first form that the user sees when accessing the membership system from the website. This is a design of the form:



The user enters their membership number and password into the respective text boxes. If after clicking the login button, the number and password match the records stored in the database then the user will be navigated to a menu screen. Otherwise an error message will be displayed giving a reason why they have been unable to log in. There is also a button on the screen for new members to click where they are directed to another form where they're able to enter their personal details.

Menu Form:

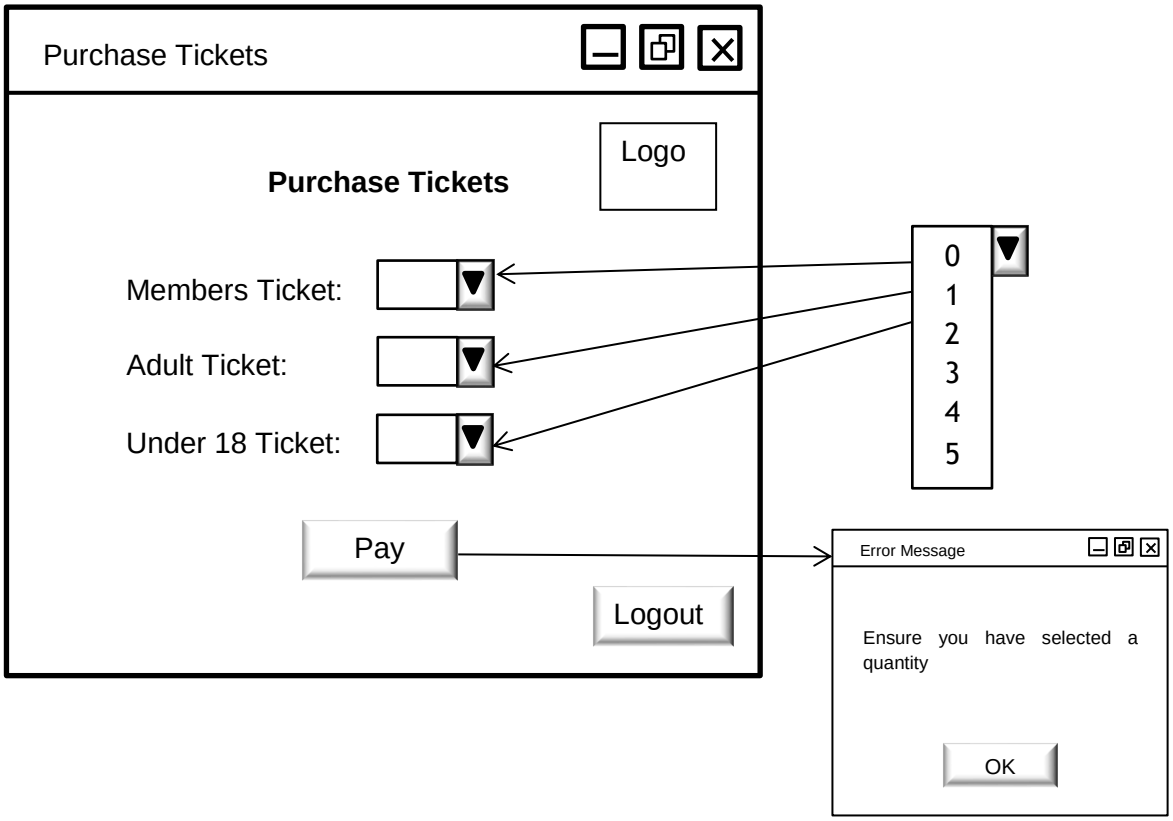
This is the menu screen that the user is directed to when the have clicked login and their membership number and password are successful:



There are three main buttons on the form, one which directs to a form where the user is able to select the tickets they want to purchase, one where they can choose the away club they're going to travel to and lastly one where they can contact the club. There is also a logout button in the bottom right hand corner which will direct the user back to the home screen when it is clicked on.

Ticket Purchase Form:

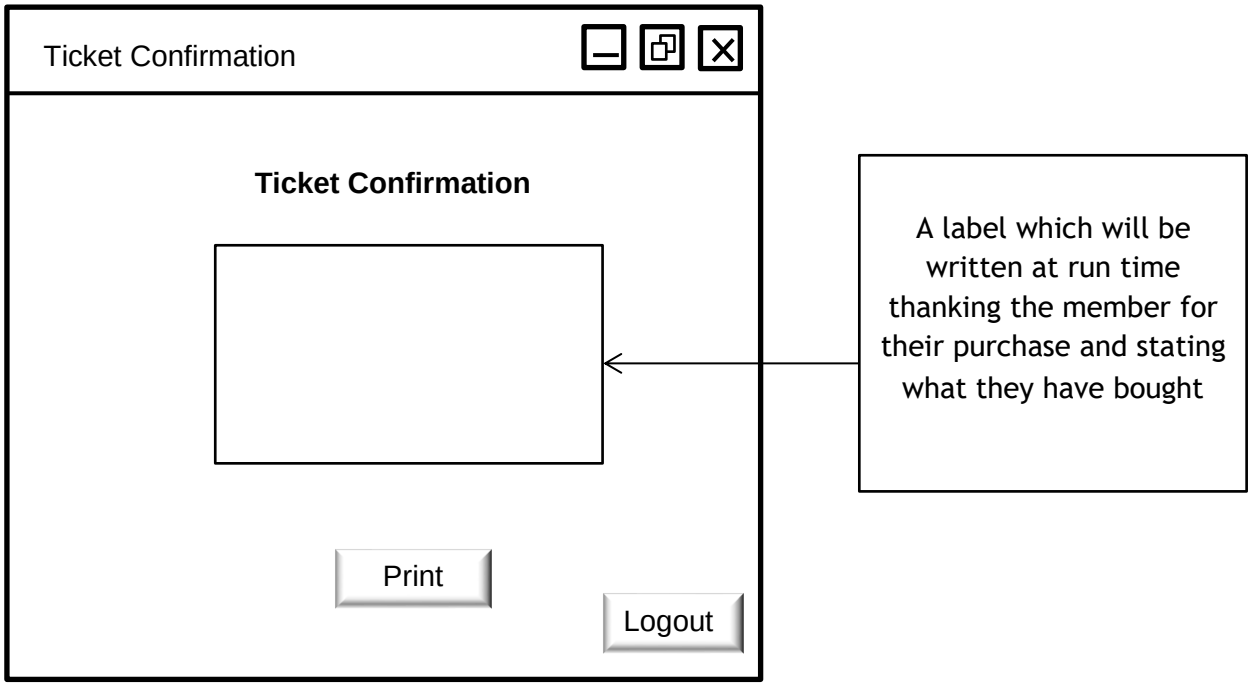
From clicking the 'Purchase Tickets' button on the menu form, the user will be directed to the form where they are able to select the tickets they want to buy.



Next to each ticket category label, there is a combination box where the user can select the quantity of that ticket type they want to purchase. When the pay button has been clicked, they will be directed to an external PayPal page where they can pay for the tickets.

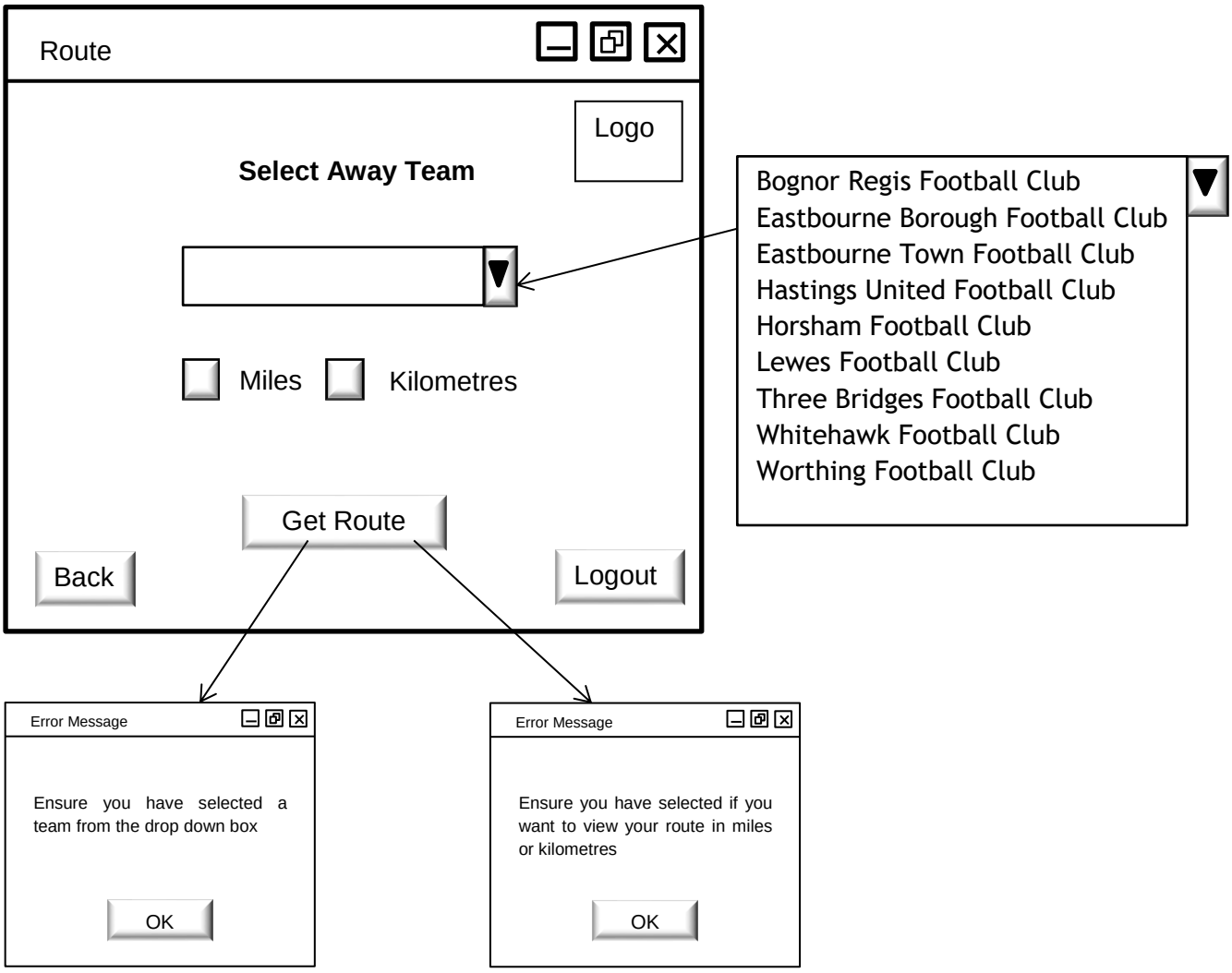
Tickets Confirmation Form:

When the user has paid for the tickets, a ticket confirmation form will pop up. The form will contain text thanking them for the purchase and confirming the tickets they have bought. They can click on the print button on the form to get a paper copy of the confirmation to bring to the match or if they are using a phone or tablet, they can screenshot the page and take it.



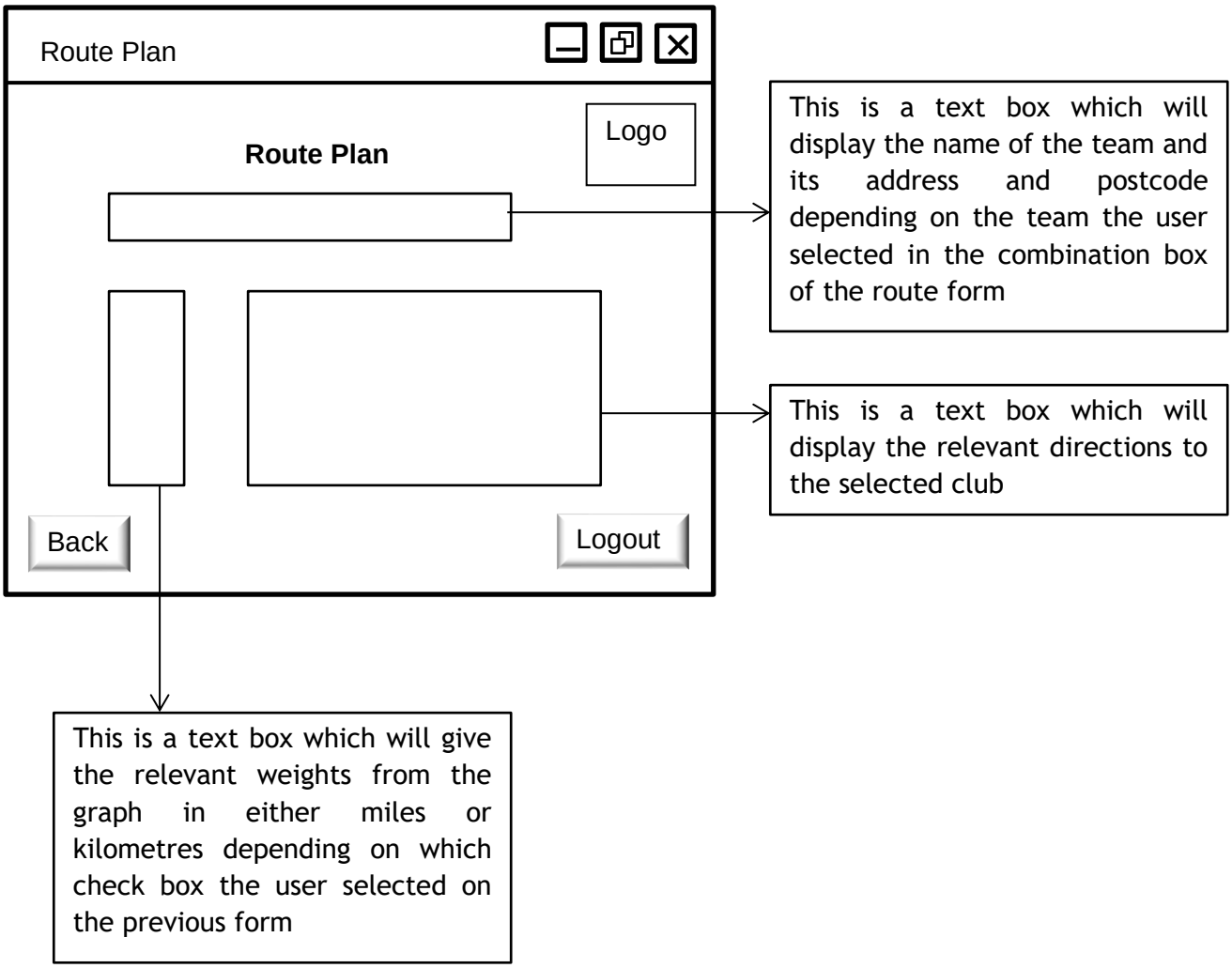
Route Form:

This form is accessed from the menu and enables the member to choose an away team in the Ryman Youth league that they want to travel to (the functionality of this can be extended to the away teams in the league of the first team in the future). There is a combination box containing the names of all the teams in the league and two check boxes depending on whether the member wants to view the directions in miles or kilometres. Then, when a team has been selected from the combo box and one of the check boxes has been ticked, a button which the user can click on when they've selected a team to be shown the route. This button links to a separate route confirmation form.



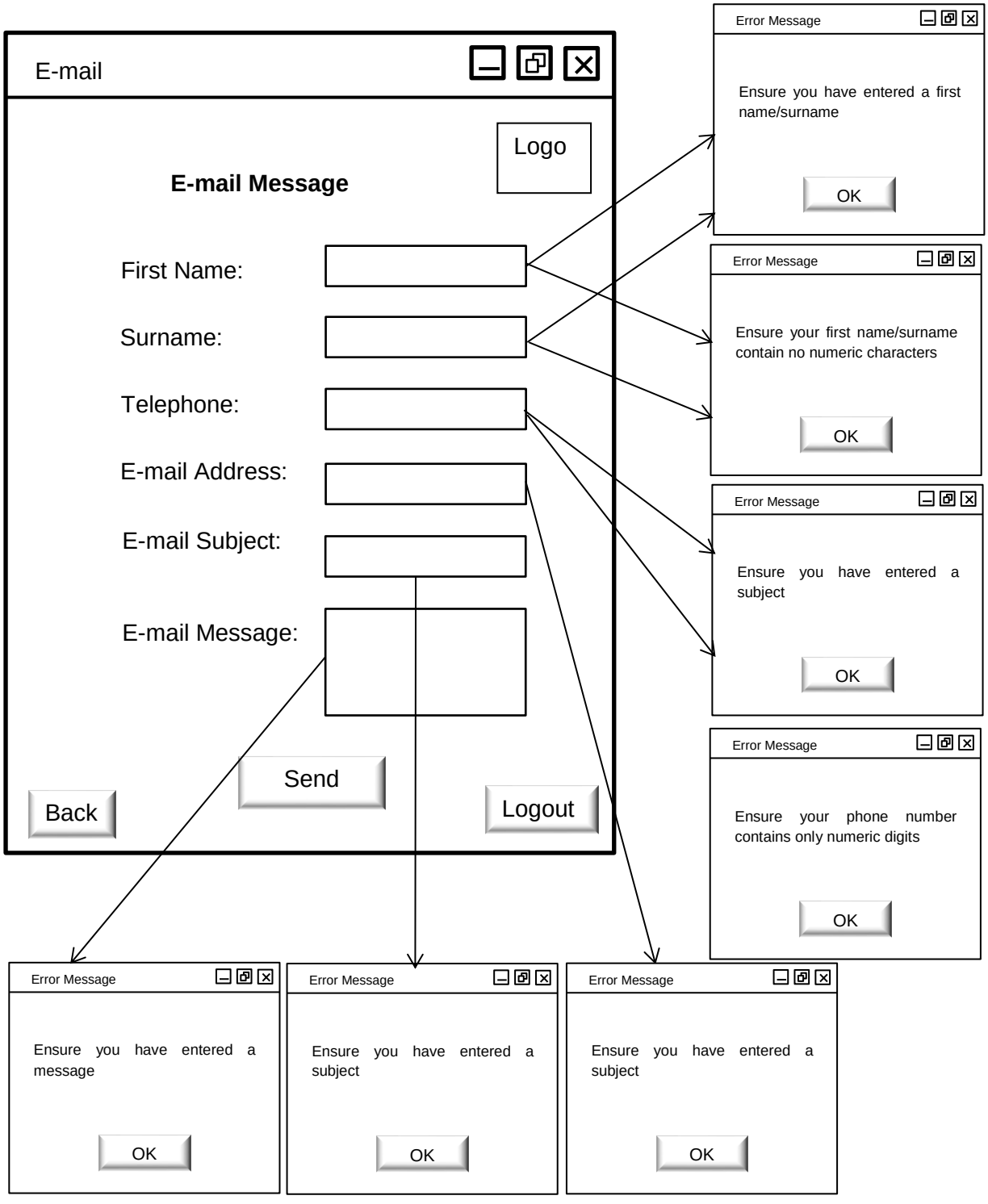
Route Plan Form:

Once the 'Get Route' button on the Route Planner form has been clicked and the algorithm for the chosen team has been calculated successfully, this form displays all the details for the route to the away team's ground.



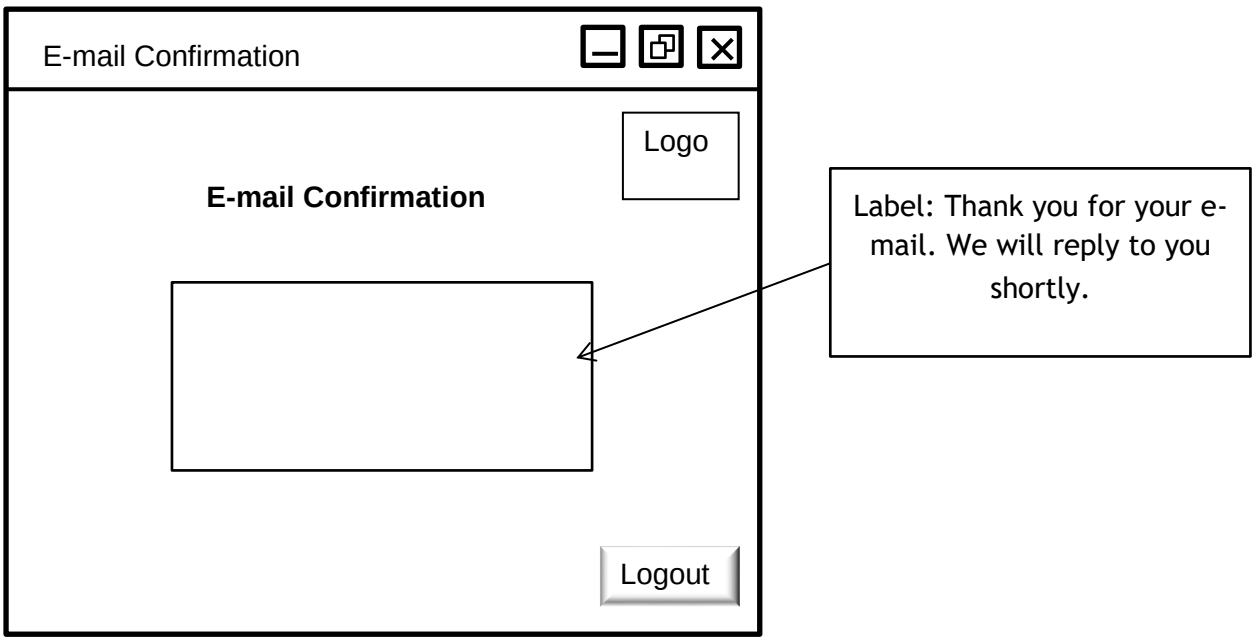
E-mail Form:

If the user has selected the contact club button on the menu, they will be directed to an e-mail form where they will be able to enter their details with an email subject and a message to send to the club. After they have finished filling the text boxes in, they can click on the send button to send the email and a confirmation form will appear if the message was successfully sent, otherwise an error message will appear informing them of why the email was unable to send.



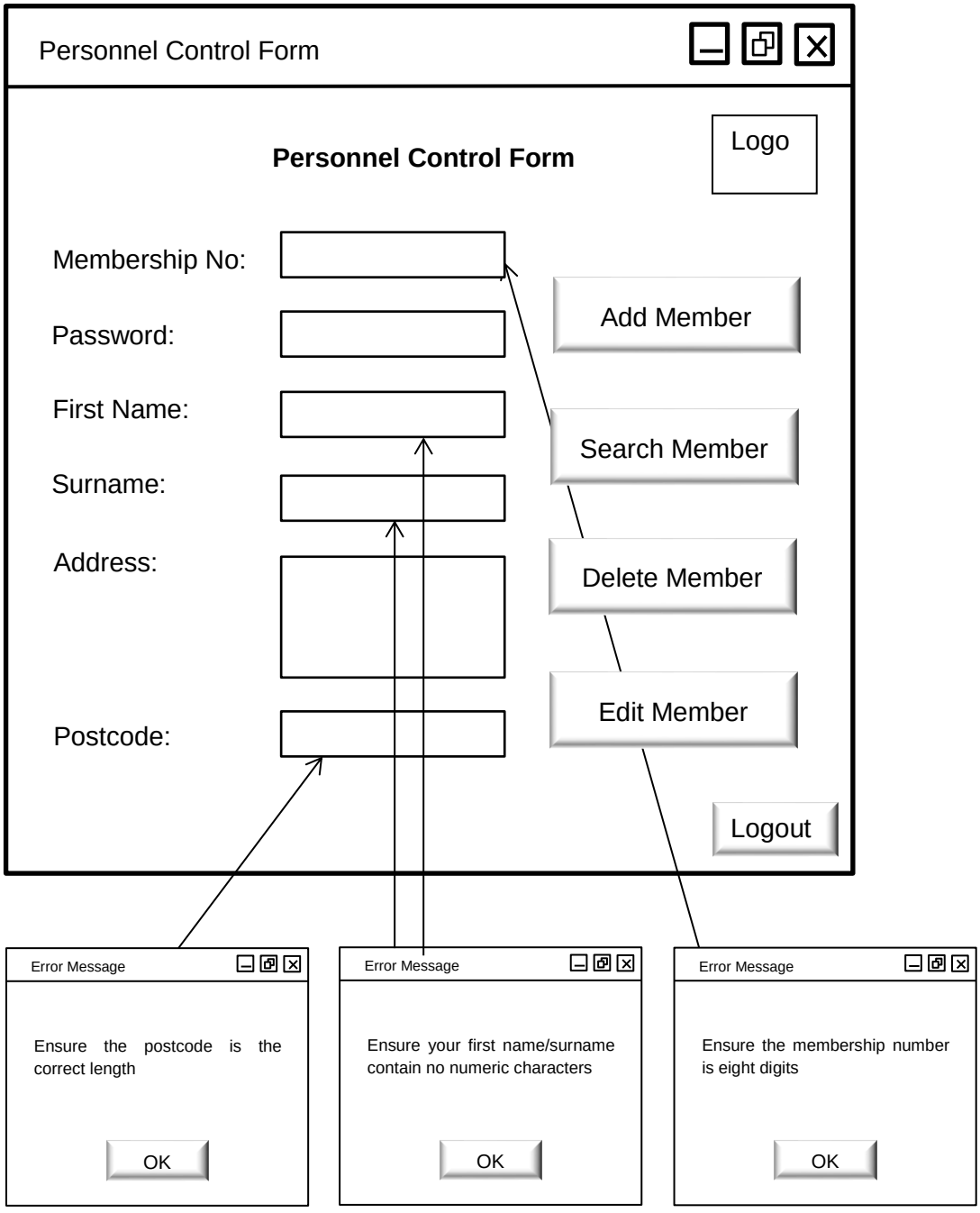
E-mail Confirmation Form:

Once the user's e-mail has been successfully sent, a confirmation screen will open thanking them for their email. After reading the confirmation, they can log out.



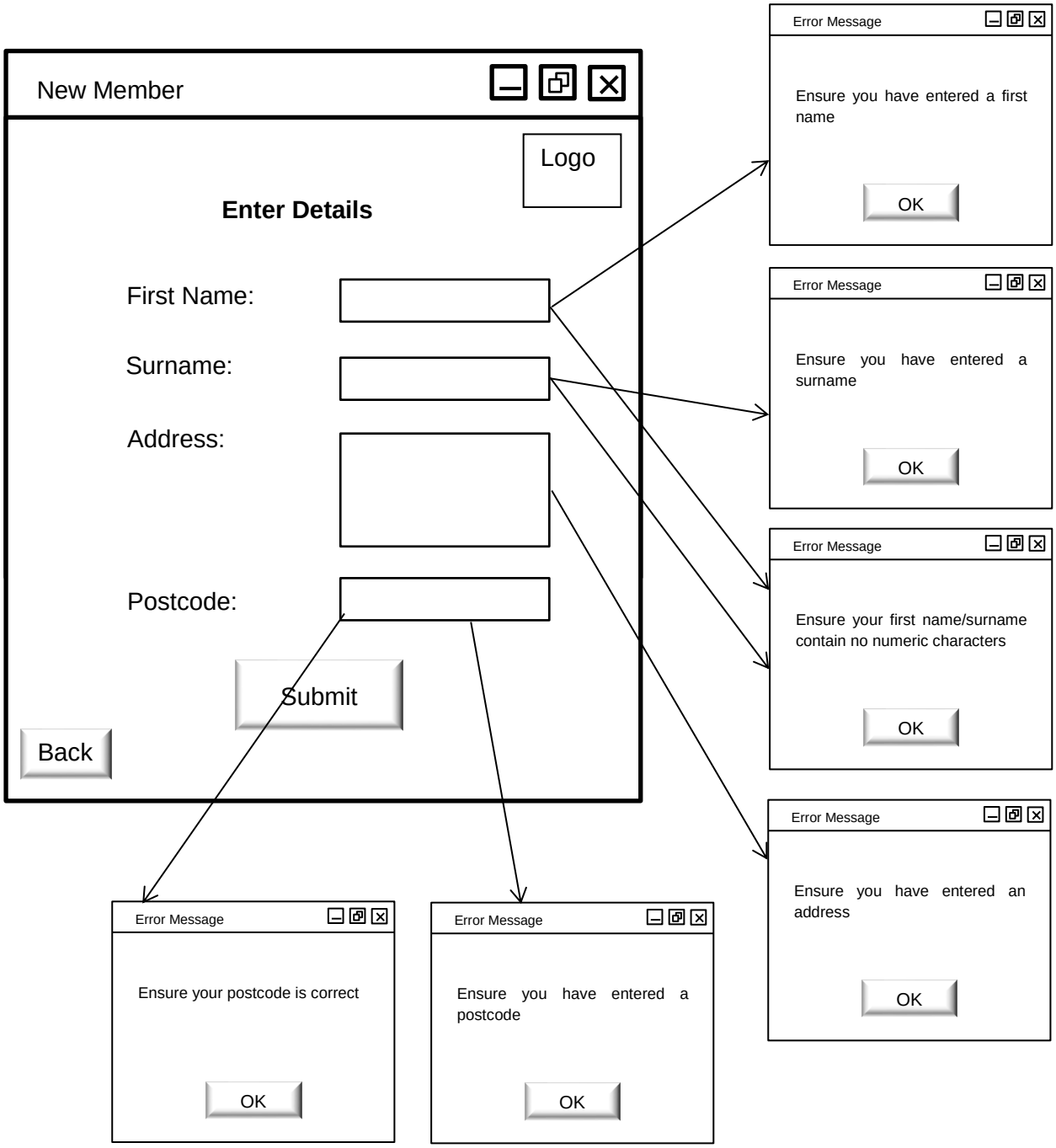
Personnel Control Form:

If a staff login number is entered on the home screen with a relevant password, they will be directed to the personnel control form. On the form there are four buttons where they are able to add, edit, delete and search members. There is a text box for each individual field in the members' table.



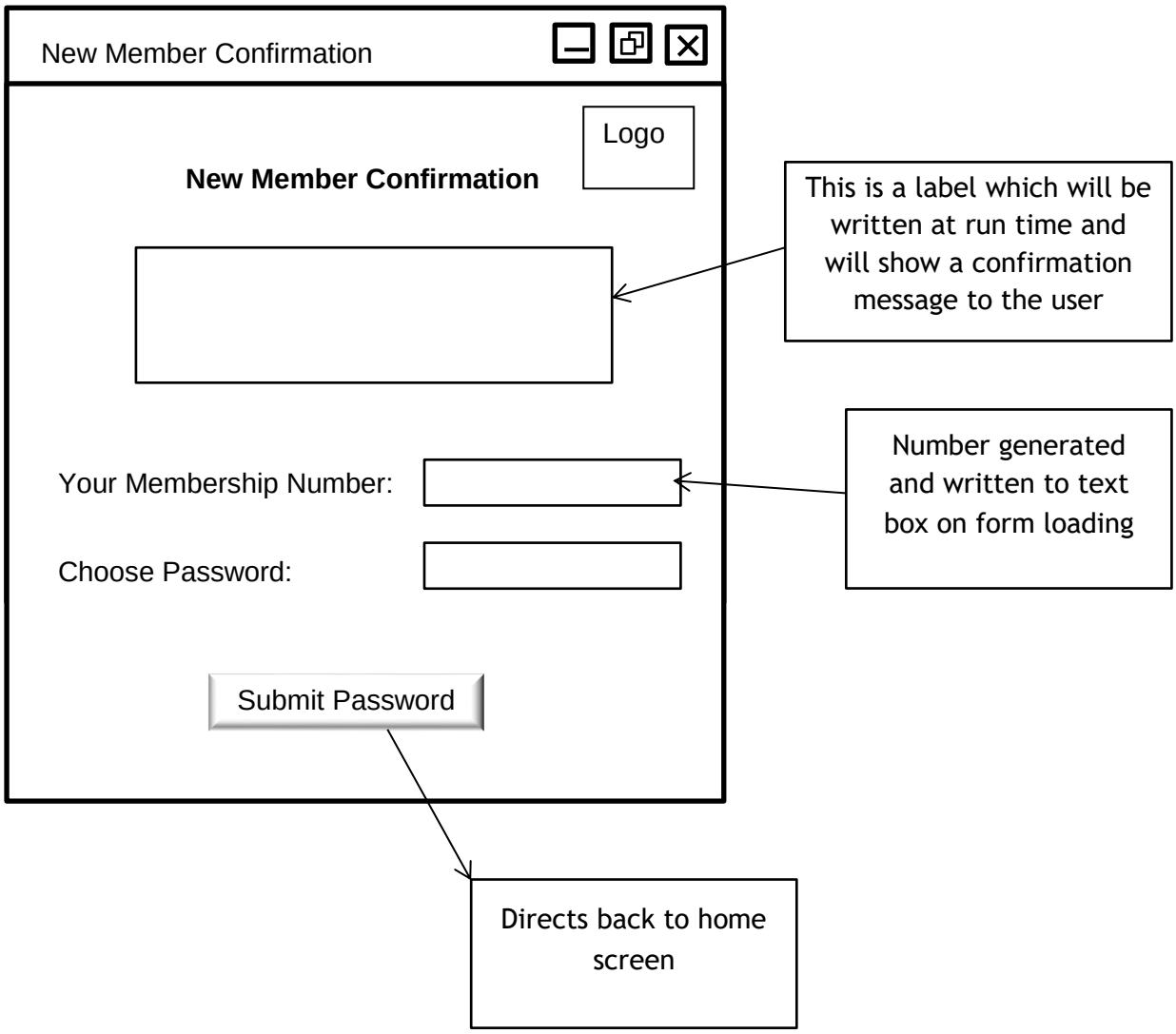
New Member Form:

If the system user is not already a user and they have clicked on the 'New Member' button on the home screen, they are directed to a form with various text boxes where they are able to fill in their basic details. After, the 'Submit' button can be clicked and if the data entered is all valid, a new row will be added to the members' database with their details. Further to this, they will be shown a confirmation screen afterwards.



New Member Confirmation Form:

When all the data filled in by the new member on the new member form is validated by the system, they will be shown a confirmation form informing them that their details have been added to the database. Also, a membership number will be generated for them and displayed in the first text box and they will have to enter a password of their choice into the second box and click the submit password button so that these can be added to the database as well. When they click the button, they will be taken back to the main home screen and will then be able to logon to the system with their membership number and password.



Form Design:

The forms in the system will need to resemble the design scheme of the club website. The club colours are green and white and the official website uses a green background with white text.

This is a design of the home screen with a theme corresponding to the one used on the club website. The background is green and all the labels are in white text, the font that has been used is Arial, the same that is used on the site. The font that appears in the text boxes when the user fills them in is also Arial and the club logo is positioned in the top right hand corner. This scheme will be consistent throughout all the forms in the system to show a high level of professionalism and make it as easy as possible for both the members and personnel to use.

Security and Data Integrity Measures:

Security Measures:

All members registered with the system will have a unique membership number and password to log on with to prevent anonymous users gaining access to the system. Likewise, all the club personnel using the system will have a login number and their own password to access the personnel control form meaning they will be the only people who are able to handle and are responsible for the stored data of the members. The passwords will be stored in the database in an encrypted format to enhance security and reduce the chances of any users account being hacked into.

Data Integrity Measures:

The system will use a range of validation algorithms to ensure the data used within the program doesn't corrupt. For example, when either a club member or member of staff initially logs on to the system, their membership or login number will be checked to ensure that it is completely numeric, it contains the right number of digits depending on the person logging in and also that it exists within the context of the database tables. If this data is not of the correct format, an error message will be displayed giving the reason for which the data is invalid. Referential integrity will need to be maintained in the system database. For instance, if a member of staff deletes a member and their details from the database using the personnel control form, the ticket records of that member will need to be simultaneously deleted to ensure that the membership number field in the tickets table is not left without a key to depend on.

System Security Measures:

To protect system security, it will be recommended that staff store the database containing all the members' data and personnel logins in a secure location on their machine. In the case of the system becoming corrupted or getting lost, copies of all the system files will be backed up on the initial USB flash drive used to install the system. The device will need to be kept in a secure physical location at the club to prevent somebody without authorisation gaining possession of it. However, as the club is relatively small and there is only a small team of staff, securing the system should not be a major problem and large threats will be unlikely to be posed.

Overall Test Strategy:

In order to ensure the membership system functions how it is intended to, I will carry out a series of tests. These will include checking that inputs are handled correctly and relevant outputs are given to the user, all the forms in the system navigate how they should, all algorithms have been implemented successfully and the database is functional.

All the tests I will initially carry out in the testing stage will be black box tests because I only want to know that particular data inputs are giving the right results. However, if any of these tests are failed, I will resort to carry out white box tests and examining all the statements in the code whilst testing all the possible combinations for each statement to ensure the problem is identified.

Input and Output Testing Design:

Firstly, I am going to test whether when data is inputted into the system, the correct outputs are returned such as the correct form being opened or the right error message is displayed to the user.

I/O Test Plan:

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments	Cross-Reference
1	A membership number and password belonging to a member are entered into the membership number and password boxes	Typical	The menu form is opened when an existing member inputs their number and password		
		Erroneous	If the data entered isn't present in the database, an error message will be shown		
		Extreme	If the membership number entered isn't of a valid format, a relevant error message will be shown		

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments	Cross-Reference
2	A login number and password belonging to a staff member are entered into the membership number and password boxes	Typical	The personnel login form is opened when a member of staff inputs their login number and password		
		Erroneous	If the data entered isn't in the personnel table of the database, an error message will be shown		
		Extreme	If the login number entered by the staff member isn't valid, an error message will be shown		
3	Data is entered into the text boxes of the new member form	Typical	A confirmation is given when all data is valid		
		Erroneous	Example: First name is entered as a number		
		Extreme	Example: Postcode contains 8 characters as opposed to 7		
4	Data is entered into the text boxes of the e-mail message form	Typical	An e-mail confirmation is given when all data is valid		
		Erroneous	Example: Telephone number contains letters		

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments	Cross-Reference
		Extreme	E-mail subject exceeds 40 characters		
5	A club is selected from the combo box of the route form	Typical	A club is selected from the list of league clubs		
			If a club isn't selected, there will be an error message informing the user they are still required to select a club		
6	The user selects if they want their route in miles or kilometres	Typical	One of the options is selected		
		Erroneous	If neither of the two options are selected, an error message is displayed		
7	Ticket quantities are selected on the ticket purchase form	Typical	Quantities are selected from all the dropdown menus		
		Erroneous	If none of the options in all or one of the combination boxes have been clicked, an error message will appear		
8	Data is entered into the personnel control form	Typical	If data entered is valid, the necessary function will be performed		

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments	Cross-Reference
		Erroneous	Example: First Name contains numeric characters, there is an error message		
		Extreme	Membership number exceeds eight characters, there is an error message		
			Postcode exceeds 7 characters, there is an error message		
9	A membership number is displayed in the text box of the member confirmation form	Typical	A membership number appears in the text box when the confirmation form is loaded		
10	Directions and distances are displayed in the text boxes of the route plan form	Typical	Directions and distances are displayed in the relevant text boxes on the form and are clear to read		

Navigation Testing Design:

The navigation of the system will need to be tested to ensure that all the forms are linked together correctly by the buttons.

All the data types in my navigation testing plan will be typical because there is only one possible option to test: if a particular button links to the form it is supposed to and it opens properly.

Navigation Test Plan:

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments	Cross-Reference
1	Home screen to members menu	Typical	The login button will open a menu form when the login button is clicked by a member		
2	Home screen to personnel control form	Typical	The login button will open the personnel control form when the login button is clicked by a member of staff		
3	Home screen to new members page	Typical	When the new members button is clicked, it will go the new members form		
4	Menu to purchase tickets page	Typical	The purchase tickets button on the menu will open the appropriate form when clicked on		
5	Menu to route planner page	Typical	The plan route button on the menu will open the appropriate form when clicked on		

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments	Cross-Reference
6	Menu to E-mail Page	Typical	The e-mail club button on the menu will open the appropriate form when clicked on		
7	Menu logout button	Typical	When the logout button is pressed, the home screen will open		
8	Purchase Tickets Page to PayPal page	Typical	When the pay button is clicked, the user will go to an external PayPal page		
9	Purchase Tickets logout button	Typical	When the logout button is pressed, the home screen will open		
10	Purchase tickets back button	Typical	When the back button is clicked, the user will be returned to the menu		
11	PayPal page to ticket confirmation page	Typical	A ticket confirmation form will open when the member has paid for their tickets		
12	Ticket Confirmation logout button	Typical	When the logout button is pressed, the home screen will open		
13	Route to route plan form	Typical	When the get route button has been clicked, the member will be given their route plan		

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments	Cross-Reference
14	Route logout button	Typical	When the logout button is pressed, the home screen will open		
15	Route back button	Typical	When the back button is clicked, the user will be returned to the menu		
16	Route plan logout button	Typical	When the logout button is pressed, the home screen will open		
17	Route plan back button	Typical	When the back button is clicked, the user will be returned to the form where they can select their route		
18	E-mail page to E-mail Confirmation	Typical	When the user has sent their e-mail, the e-mail confirmation form will open		
19	E-mail logout button	Typical	When the logout button is pressed, the home screen will open		
20	E-mail back button	Typical	When the back button is clicked, the user will be returned to the menu		
21	E-mail confirmation logout button	Typical	When the logout button is pressed, the home screen will open		

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments	Cross-Reference
22	New Member to new member confirmation	Typical	When a new member has submitted their details, they will get a confirmation form		
23	New member back button	Typical	When the back button is clicked, the user will be returned to the home screen		
24	New member confirmation to home screen	Typical	When the user has submitted their password, the home screen will be accessed		

Algorithm Testing Design:

I will also be testing the system to see if all the algorithms I have implemented function how they're supposed to and/or give the intended results. As part of the algorithm testing, I will also be tracing the more complex of the algorithms in tables to see the results they produce step by step.

Algorithm Test Plan:

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments	Cross-Reference
1	Membership Number Validation	Typical	If a membership number is valid, the menu form can be accessed by the user		
		Erroneous	If the membership number doesn't exist, an appropriate error message will be given to the user		
		Extreme	If the membership number isn't the correct length, an error message will be shown		
2	Member's Password Validation	Typical	If the member's password is valid, the menu form can be accessed by the user		

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments	Cross-Reference
		Erroneous	If the password doesn't match the same member as the membership number or doesn't exist, an error message will be shown		
3	Login Number Validation	Typical	If a login number is valid, the personnel control form can be accessed by the personnel		
		Erroneous	If the login number doesn't exist, an appropriate error message will be given to the staff member		
			If the login number isn't the correct length, an error message will be shown		
4	Staff Password Validation	Typical	If the staff member's password is valid, the personnel control form can be accessed by the user		

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments	Cross-Reference
		Erroneous	If the password doesn't match the same staff member as the login number or doesn't exist, an error message will be shown		
5	Membership Number Generation	Typical	A membership number will be generated that is the lowest possible eight digit number, not currently present in the members' table		
6	Calculate Ticket Prices	Typical	The correct total will be calculated with the ticket options that the user has selected		
7	Route Planner	Typical	The user will be given the route to the correct club, with the directions being in consecutive order, the distances being in the measure the user originally selected and the directions and distances corresponding		

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments	Cross-Reference
8	Password Encryption	Typical	In the database, an encrypted form of the member's password will be stored. When the user logs on, their password will be encrypted and then compared to the database encryption		

Functionality Testing Design:

It is important that all the features on the forms work correctly for the system user such as the text boxes, the combination boxes and the labels. I will not test most of the buttons in the functionality testing because I will be testing most of these functions within the navigation testing.

Functionality Test Plan:

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments?	Cross-Reference
1	Text can be entered into the membership number and password text boxes on the home screen	Typical	Text can be written in both boxes before the user logs in		
2	Text can be entered in all the boxes of the personnel control form	Typical	Text can be written in the boxes for members data		
3	When the search member button on the personnel control form are clicked, text appears in every box	Typical	When a member is searched on the form, the members data from the database appears in the textboxes		
4	Text can be entered in all the boxes of the new member form	Typical	A new member is able to input all their data into the boxes on the new member form		
5	All the combination boxes on the ticket purchase function properly	Typical	The three dropdown menus on the ticket purchasing form all contain the options 0 to 5		

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments?	Cross-Reference
6	The combination box on the route form functions properly	Typical	The box drops down and shows all the league teams		
7	Text can be entered in all of the text boxes in the e-mail form	Typical	A member is able to input into every box on the e-mail form		
8	The label on the new member confirmation form outputs text when the form is opened	Typical	There is a label displayed on the confirmation form		
9	The membership number text box outputs a membership number for a new member	Typical	A membership number is written in the text box		
10	The user can enter text into the password text box on the new member confirmation form	Typical	Text can be entered into the password box		
11	The label on the ticket confirmation form outputs text when the form is opened	Typical	There is a label displayed showing the tickets bought		
12	The text box on the route plan form outputs the route	Typical	All text boxes on the route form have read in data and displayed it		

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments?	Cross-Reference
13	The label on the e-mail confirmation form outputs text when it is opened	Typical	There is a label displayed giving an e-mail confirmation		

Database Testing Design:

The database testing will comprise of tests which see whether the database functions work how they're supposed to. For instance, to test whether members are successfully added to and deleted from the member's database.

Database Test Plan:

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments?	Cross-Reference
1	Adding member to database on personnel control form	Typical	When data in all boxes is valid, the member will be added to the members table		
2	Adding new member when a new member enters their details	Typical	When all the details the new member has filled in have been validated, they will be added to the database		
3	New membership number and password are added to new member's record	Typical	When the user clicks on submit password, the number that has been generated for the member and the password will be added to the database		
4	Searching for a particular member in the personnel control form	Typical	When one of the member's details is entered into one of the text boxes on the control form and the search button is pressed, the remaining details will appear in the other boxes		

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments?	Cross-Reference
5	Deleting a member from the database	Typical	When data in all boxes is valid, the member will be deleted from the members table		
6	Updating the details of a particular member	Typical	When all the boxes have been filled, a members details will be overwritten by the current data		

Testing:

I will now use the test plans I designed in the design section to test the various aspects of the system. For each test, I will compare what the expected outcome is to the actual outcome of the test to see if the test has passed or failed. If there is a failed outcome, I will have to take the test further

Input and Output Testing:

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments	Cross-Reference
1	A membership number and password belonging to a member are entered into the membership number and password boxes	Typical	The menu form is opened when an existing member inputs their number and password	Pass When a membership number and password are entered that exist in the database, the menu form is opened	1
		Erroneous	If the data entered isn't present in the database, an error message will be shown	Pass When a membership is entered that is not in the database, an error message appears. Also, if the password is not recognised, an error is displayed	2, 3
			If there is no membership number or no password entered, there will be an appropriate error message	Pass If any of the boxes are left without text, an error message will appear	4, 5

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments	Cross-Reference
		Extreme	If the membership number entered isn't of a valid format, a relevant error message will be shown	Pass When a nine digit membership number is entered, there is an error	6
2	A login number and password belonging to a staff member are entered into the membership number and password boxes	Typical	The personnel login form is opened when a member of staff inputs their login number and password	Pass When a staff login number and password are entered that exist in the database, the personnel control form is opened	7
		Erroneous	If the data entered isn't in the personnel table of the database, an error message will be shown	Pass When a login number is entered that doesn't exist, an error is shown When a password is entered that doesn't exist, there is an error	8,9
			If there is no login number or no password entered, there will be an appropriate error message	Pass If one of the boxes on the form are left empty, there is an error	10,11

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments	Cross-Reference
		Extreme	If the login number entered by the staff member isn't valid, an error message will be shown	Pass When a six digit number is entered, there is an error	12
3	Data is entered into the text boxes of the new member form	Typical	A confirmation is given when all data is valid	Pass The confirmation form is shown	13
		Erroneous	Example: First name is entered as a number	Pass When there is a number in the name, there is an error when the data is submitted	14
			If no data is entered into one of the boxes, there will be an error	Pass If at least one of the fields is empty, there is an error	15
		Extreme	Example: Postcode contains 8 characters as opposed to 7	Pass If the postcode is not the standard length, there is an error	16
4	Data is entered into the text boxes of the e-mail message form	Typical	An e-mail confirmation is given when all data is valid	Pass When all the data is valid, the user receives a confirmation	17
		Erroneous	Example: Telephone number contains letters	Pass When the number contains letters, there is a relevant error message informing the user	18

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments	Cross-Reference
		Extreme	E-mail subject exceeds 40 characters	Pass If the subject is more than 40 characters long, there is an error	19
5	A club is selected from the combo box of the route form	Typical	A club is selected from the list of league clubs	Pass A club can be selected from the dropdown menu	20
			If a club isn't selected, there will be an error message informing the user they are still required to select a club	Pass When the 'Get Route' button is clicked, there is an error if the dropdown menu has been left containing no team	21
6	The user selects if they want their route in miles or kilometres	Typical	One of the options is selected	Pass The miles or kilometres checkboxes can be checked	22
		Erroneous	If neither of the two options are selected, an error message is displayed	Fail There is an error displayed when neither of the options are selected but not when both are selected at the same time	23

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments	Cross-Reference
7	Ticket quantities are selected on the ticket purchase form	Typical	Quantities are selected from all the dropdown menus	Pass Quantities can be selected from each of the ticket menus and once the 'Pay' button is clicked, the user is directed to the PayPal page	24
		Erroneous	If none of the options in all or one of the combination boxes have been clicked, an error message will appear	Pass There is a separate error message for each of the menus which is left unselected	25,26,27
8	Data is entered into the personnel control form	Typical	If data entered is valid, the necessary function will be performed	Pass When entering a valid membership number to search for a member, the member details are returned	28
		Erroneous	Example: First Name contains numeric characters, there is an error message	Pass When a name is entered containing a number and the add member button is clicked, there is an error	29
		Extreme	Membership number exceeds eight	Pass When a nine digit membership	30

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments	Cross-Reference
			characters, there is an error message	number is entered into the membership number box on the form, there is an error	
			Postcode exceeds 7 characters, there is an error message	Pass If the postcode is eight characters as opposed to 7, there is an error	31
9	A membership number is displayed in the text box of the member confirmation form	Typical	A membership number appears in the text box when the confirmation form is loaded	Pass There is a membership number written to the text box when the form loads	32
10	Directions and distances are displayed in the text boxes of the route plan form	Typical	Directions and distances are displayed in the relevant text boxes on the form and are clear to read	Pass Directions and distances are displayed in the text boxes of the route plan form when it loads	33

Navigation Testing:

I will not provide screenshots for navigation testing because there will only be one outcome to test and all data types are typical

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments	Cross-Reference
1	Home screen to members menu	Typical	The login button will open a menu form when the login button is clicked by a member	Pass	-
2	Home screen to personnel control form	Typical	The login button will open the personnel control form when the login button is clicked by a member of staff	Pass	-
3	Home screen to new members page	Typical	When the new members button is clicked, it will go the new members form	Pass	-
4	Menu to purchase tickets page	Typical	The purchase tickets button on the menu will open the appropriate form when clicked on	Pass	-
5	Menu to route planner page	Typical	The plan route button on the menu will open the appropriate form when clicked on	Pass	-
6	Menu to E-mail Page	Typical	The e-mail club button on the menu will open the appropriate	Pass	-

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments	Cross-Reference
			form when clicked on		
7	Menu logout button	Typical	When the logout button is pressed, the home screen will open	Pass	-
8	Purchase Tickets Page to PayPal page	Typical	When the pay button is clicked, the user will go to an external PayPal page	Pass	-
9	Purchase Tickets logout button	Typical	When the logout button is pressed, the home screen will open	Pass	-
10	Purchase tickets back button	Typical	When the back button is clicked, the user will be returned to the menu	Pass	-
11	PayPal page to ticket confirmation page	Typical	A ticket confirmation form will open when the member has paid for their tickets	Pass	-
12	Ticket Confirmation logout button	Typical	When the logout button is pressed, the home screen will open	Pass	-
13	Route to route plan form	Typical	When the get route button has been clicked, the member will be given their route plan	Pass	-
14	Route logout button	Typical	When the logout button is	Pass	-

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments	Cross-Reference
			pressed, the home screen will open		
15	Route back button	Typical	When the back button is clicked, the user will be returned to the menu	Pass	-
16	Route plan logout button	Typical	When the logout button is pressed, the home screen will open	Pass	-
17	Route plan back button	Typical	When the back button is clicked, the user will be returned to the form where they can select their route	Pass	-
18	E-mail page to E-mail Confirmation	Typical	When the user has sent their e-mail, the e-mail confirmation form will open	Pass	-
19	E-mail logout button	Typical	When the logout button is pressed, the home screen will open	Pass	-
20	E-mail back button	Typical	When the back button is clicked, the user will be returned to the menu	Pass	-
21	E-mail confirmation logout button	Typical	When the logout button is pressed, the home screen will open	Pass	-
22	New Member to new member	Typical	When a new member has submitted	Pass	-

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments	Cross-Reference
	confirmation		their details, they will get a confirmation form		
23	New member back button	Typical	When the back button is clicked, the user will be returned to the home screen	Pass	-
24	New member confirmation to home screen	Typical	When the user has submitted their password, the home screen will be accessed	Pass	-

Algorithm Testing:

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments	Cross-Reference
1	Membership Number Validation	Typical	If a membership number is valid, the menu form can be accessed by the user	Pass The menu form is opened if the details entered are valid and the password corresponds to the membership number	34
		Erroneous	If the membership number doesn't exist, an appropriate error message will be given to the user	Pass If a membership number is entered which does not exist in the members table, there is an error given to the user	35
		Extreme	If the membership number isn't the correct length, an error message will be shown	Pass If the membership number entered is nine digits in length, a relevant error is given to the member	36
2	Member's Password Validation	Typical	If the member's password is valid, the menu form can be accessed by the user	Pass If there is a valid password together with valid membership number, the menu form is opened	37

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments	Cross-Reference
		Erroneous	If the password doesn't match the same member as the membership number or doesn't exist, an error message will be shown	Pass If the password entered does not match the same account as the membership number belongs to, there is an error	38
3	Login Number Validation	Typical	If a login number is valid, the personnel control form can be accessed by the personnel	Pass When a valid personnel login number is entered, provided the password is valid, the personnel control form is opened	39
		Erroneous	If the login number doesn't exist, an appropriate error message will be given to the staff member	Pass If a login number is entered which doesn't exist in the personnel table, there is an error	40
			If the login number isn't the correct length, an error message will be shown	Pass If the login number entered is six characters in length, there is an error informing the user the length is wrong	41

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments	Cross-Reference
4	Staff Password Validation	Typical	If the staff member's password is valid, the personnel control form can be accessed by the user	Pass If there is a valid password together with valid login number, the personnel control form is opened	42
		Erroneous	If the password doesn't match the same staff member as the login number or doesn't exist, an error message will be shown	Pass If the password entered does not match the same account as the login number belongs to, there is an error	43
5	Membership Number Generation	Typical	A membership number will be generated that is the lowest possible eight digit number, not currently present in the members' table	Pass Although the number generated can't be added to the database, the number generated is the lowest possible that is not present in the members table	44
6	Calculate Ticket Prices	Typical	The correct total will be calculated with the ticket options that the user has selected	Pass The ticket price calculated is the quantity multiplied by the fixed prices of the different ticket categories giving the correct value	45

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments	Cross-Reference
7	Route Planner	Typical	The user will be given the route to the correct club, with the directions being in consecutive order, the distances being in the measure the user originally selected and the directions and distances corresponding	Fail Although the name of the club that is given is correct and the directions are right, the distances given in the left text box do not correspond to the right routes to the selected club	46
8	Password Encryption	Typical	In the database, an encrypted form of the member's password will be stored. When the user logs on, their password will be encrypted and then compared to the database encryption	Fail Although the system encrypts the password on both the home screen and the new member confirmation form, there is no method which adds or compares the encrypted form of the password to the database	47

Functionality Testing:

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments?	Cross-Reference
1	Text can be entered into the membership number and password text boxes on the home screen	Typical	Text can be written in both boxes before the user logs in	Pass Both boxes contain text before the user clicks on the log on button	48
2	Text can be entered in all the boxes of the personnel control form	Typical	Text can be written in the boxes for members data	Pass All boxes on the personnel control form can be written to if required	49
3	When the search member button on the personnel control form are clicked, text appears in every box	Typical	When a member is searched on the form, the members data from the database appears in the textboxes	Pass When the search member button is clicked, provided all the data is valid, text appears in the	50
4	Text can be entered in all the boxes of the new member form	Typical	A new member is able to input all their data into the boxes on the new member form	Pass All boxes on the new member form can have data inputted into them	51
5	All the combination boxes on the ticket purchase function properly	Typical	The three dropdown menus on the ticket purchasing form all contain the options 0 to 5	Pass Every dropdown menu on the ticket purchase form is able to have an item selected form it	52

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments?	Cross-Reference
6	The combination box on the route form functions properly	Typical	The box drops down and shows all the league teams	Pass All the clubs in the youth league appear in the dropdown menu on the route form, they are also all visible in the box	53
7	Text can be entered in all of the text boxes in the e-mail form	Typical	A member is able to input into every box on the e-mail form	Pass Every box on the e-mail form can have text inputted into it	54
8	The label on the new member confirmation form outputs text when the form is opened	Typical	There is a label displayed on the confirmation form	Pass The label on the new member confirmation form is written when the form is loaded	55
9	The membership number text box outputs a membership number for a new member	Typical	A membership number is written in the text box	Pass There is a membership number outputted to the text box on the confirmation form	56
10	The user can enter text into the password text box on the new member confirmation form	Typical	Text can be entered into the password box	Pass The text box where the user enters their choice of password can be written to	57

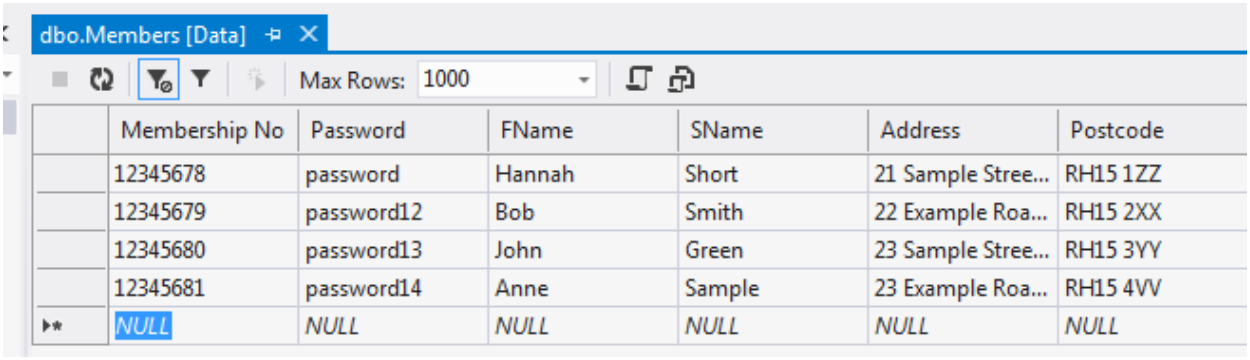
Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments?	Cross-Reference
11	The label on the ticket confirmation form outputs text when the form is opened	Typical	There is a label displayed showing the tickets bought	Pass The label on the ticket confirmation form is written when the form is opened	58
12	The text boxes on the route plan form outputs the route	Typical	All text boxes on the route form have read in data and displayed it	Fail Although the data from the text files are read into the text boxes, most of the route can't be seen on the form due to the size	59
13	The label on the e-mail confirmation form outputs text when it is opened	Typical	There is a label displayed giving an e-mail confirmation	Pass The label on the e-mail confirmation form is written when the form loads thanking the user for their message	60

Test Data Set:

In order to test my database, I will be using a set of minimal test data to model how the data in the database can be manipulated and how successive members can be added to the members table.

I will populate the members table of the database with the following test data:

Membership Number	Password	First Name	Surname	Address	Postcode
12345678	password	Hannah	Short	21 Sample Street Sample Village	RH15 1ZZ
12345679	password12	Bob	Smith	22 Example Road Example Town	RH15 2XX
12345680	password13	John	Green	23 Sample Street Sample Village	RH15 3YY
12345681	password14	Anne	Sample	23 Example Road Example Town	RH15 4VV



Membership No	Password	FName	SName	Address	Postcode
12345678	password	Hannah	Short	21 Sample Stree...	RH15 1ZZ
12345679	password12	Bob	Smith	22 Example Roa...	RH15 2XX
12345680	password13	John	Green	23 Sample Stree...	RH15 3YY
12345681	password14	Anne	Sample	23 Example Roa...	RH15 4VV
NULL	NULL	NULL	NULL	NULL	NULL

Database Testing:

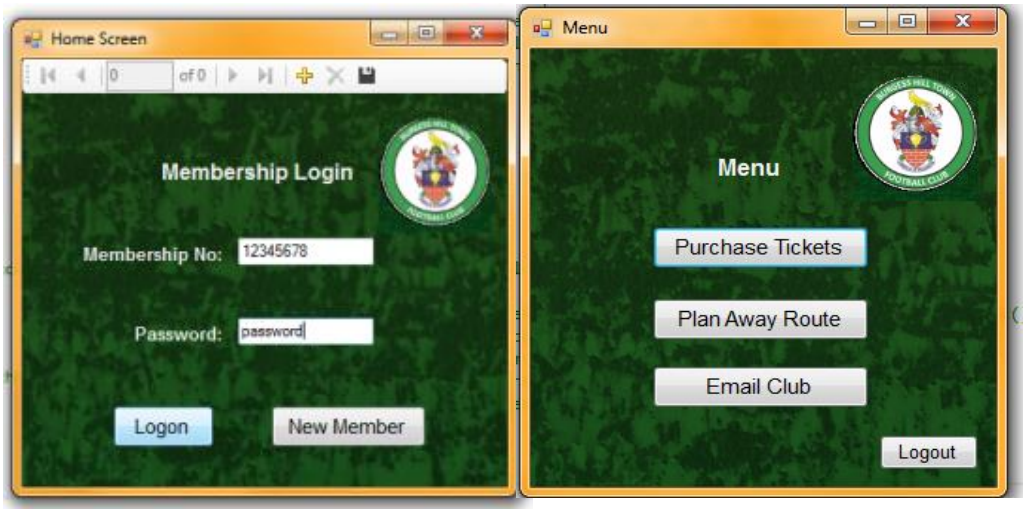
Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments?	Cross-Reference
1	Adding member to database on personnel control form	Typical	When data in all boxes is valid, the member will be added to the members table	Pass When the add member button is clicked, a member is added to the member table	61
2	Adding new member when a new member enters their details	Typical	When all the details the new member has filled in have been validated, they will be added to the database	Fail The member details can't be added without the presence of the membership number because it is the primary key of the data, to rectify this the number will have to be added at the same time as the other data and the password after	62
3	New membership number and password are added to new member's record	Typical	When the user clicks on submit password, the number that has been generated for the member and the password will be added to the database	Fail The membership number cannot be added separately to the other member details because they depend on the number as a key, it will need to	63

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments?	Cross-Reference
				be added with them and the password chosen can be added later	
4	Searching for a particular member in the personnel control form	Typical	When one of the member's details is entered into one of the text boxes on the control form and the search button is pressed, the remaining details will appear in the other boxes	Pass When the membership number of a member is entered, provided that they are present in the database, their other details are loaded into the boxes on the form	64
5	Deleting a member from the database	Typical	When data in all boxes is valid, the member will be deleted from the members table	Pass When the delete member on the personnel control form is clicked after inputting the membership number of the member who is being deleted, the member is removed from the table	65
6	Updating the details of a particular member	Typical	When all the boxes have been filled, a members details will be	Pass Members details are able to be overwritten when the	66

Test Number	Description	Data Type	Expected Result	Pass/Fail? Comments?	Cross-Reference
			overwritten by the current data	edit member button is clicked on the personnel control form	

Testing Evidence:

Input/output Testing:



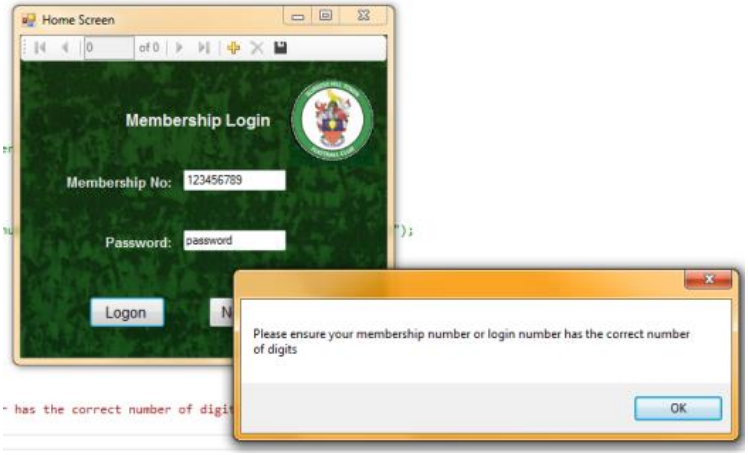
Screenshot 1: The menu form has appeared when the membership number and password have been validated

Screenshot 2: When a membership number is entered which doesn't exist, there is an error

Screenshot 3: When a password is entered that doesn't exist or does not correspond with the membership number, there is an error

Screenshot 4: No membership number has been entered into the membership number text box, there is an error message

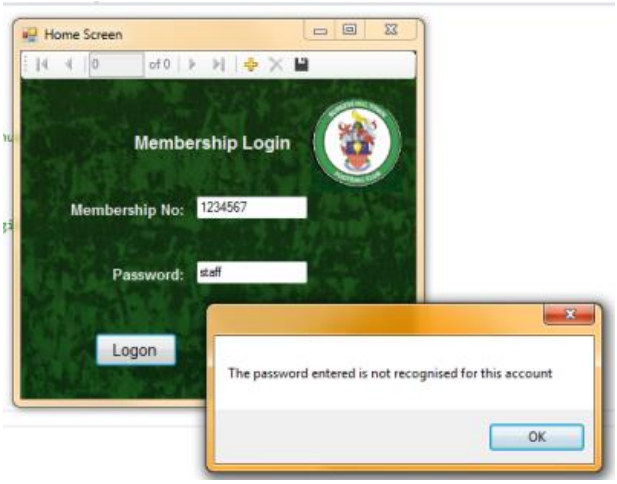
Screenshot 5: No password has been entered into the password text box, there is an error message



Screenshot 6: The membership number entered contains one too many digits, there is an error

Screenshot 7: When the login number and password have been validated, the personnel control form appears

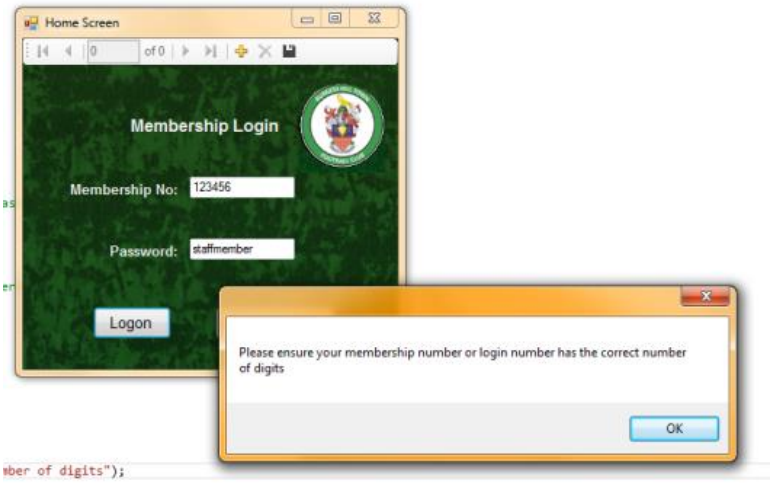
Screenshot 8: When a login number is entered that doesn't exist in the personnel table, there is an error



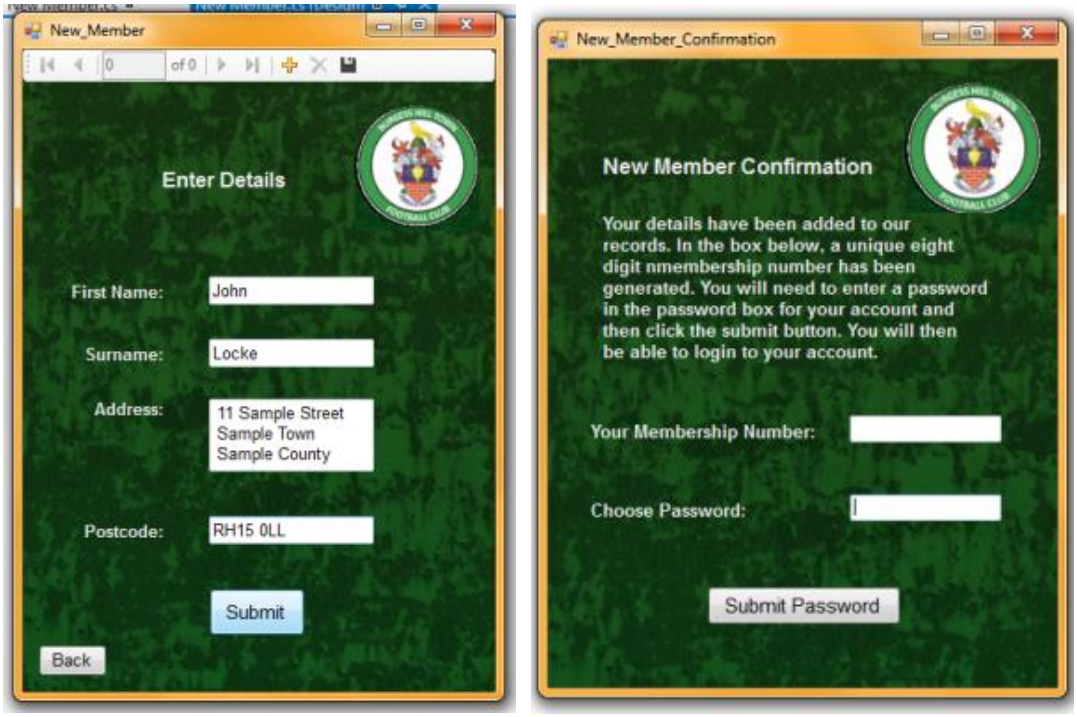
Screenshot 9: When a password is entered that doesn't exist in the personnel table or does not correspond to the same account as the login number, there is an error

Screenshot 10: No login number has been entered into the membership text box, there is an error message

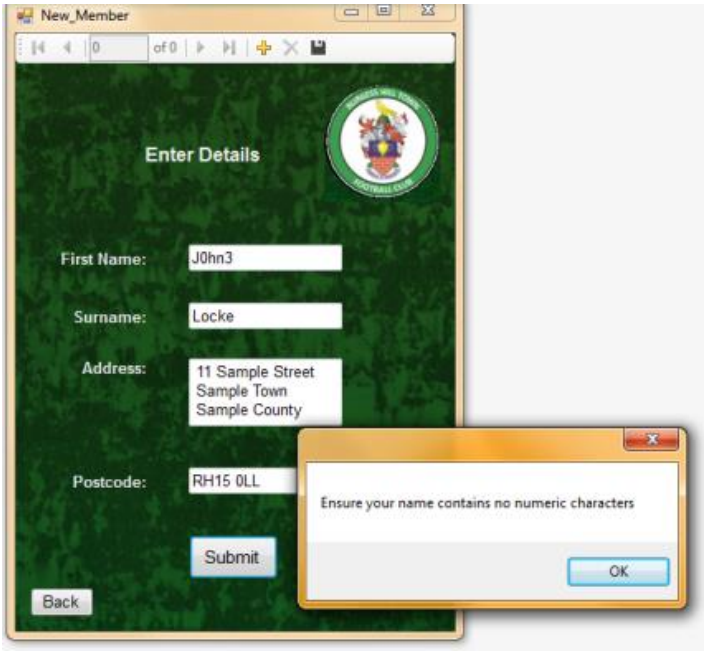
Screenshot 11: No password has been entered into the password text box, there is an error message



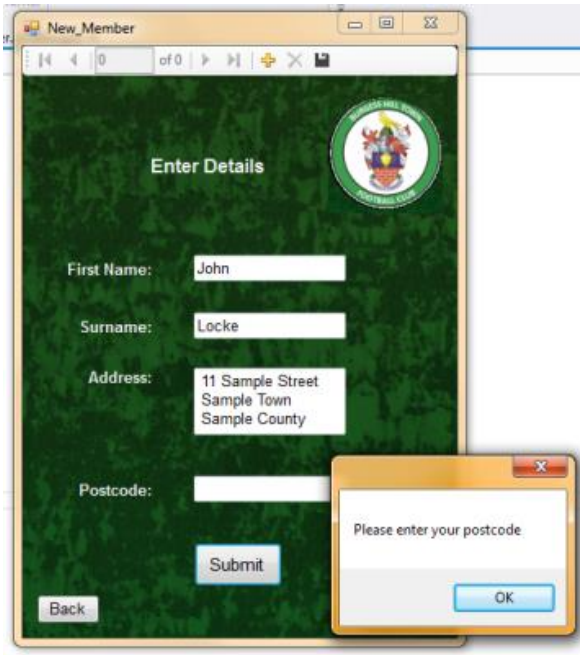
Screenshot 12: If a login number is entered containing too few digits, there is an error



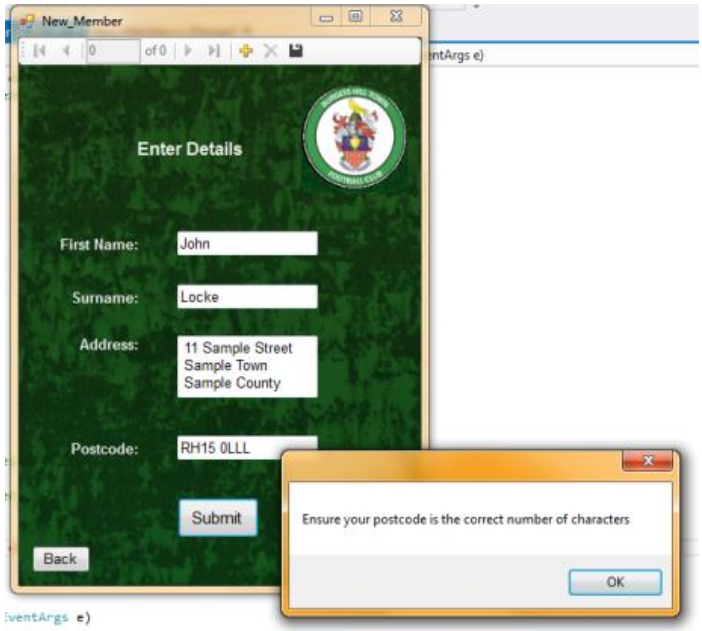
Screenshot 13: When valid data is entered, the new member confirmation appears



Screenshot 14: If a name is entered containing a number, there is an error message

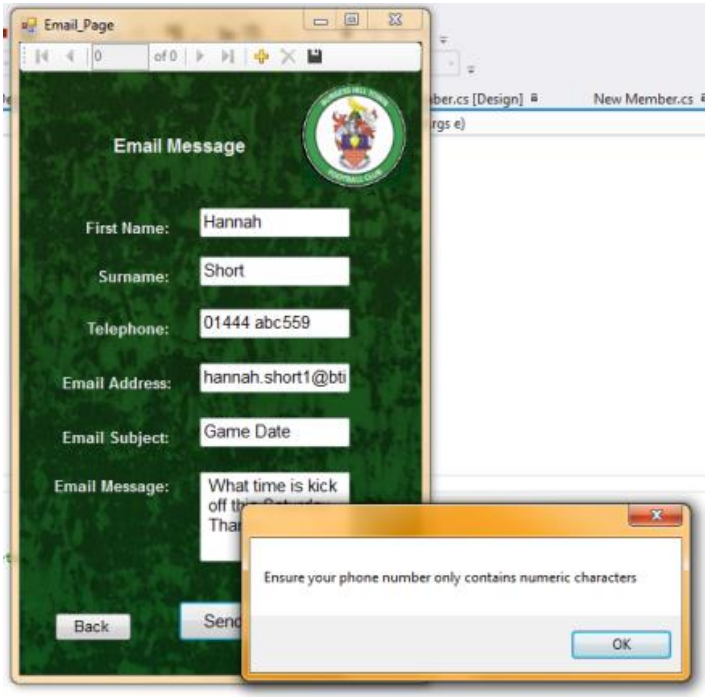


Screenshot 15: When a field is left blank, there is an error message



Screenshot 16: When a postcode with the wrong number of characters is entered, there is an error

Screenshot 17: When the data inputted to the form is valid, the confirmation form is shown



Screenshot 18: If a phone number containing letters is entered, there is an error

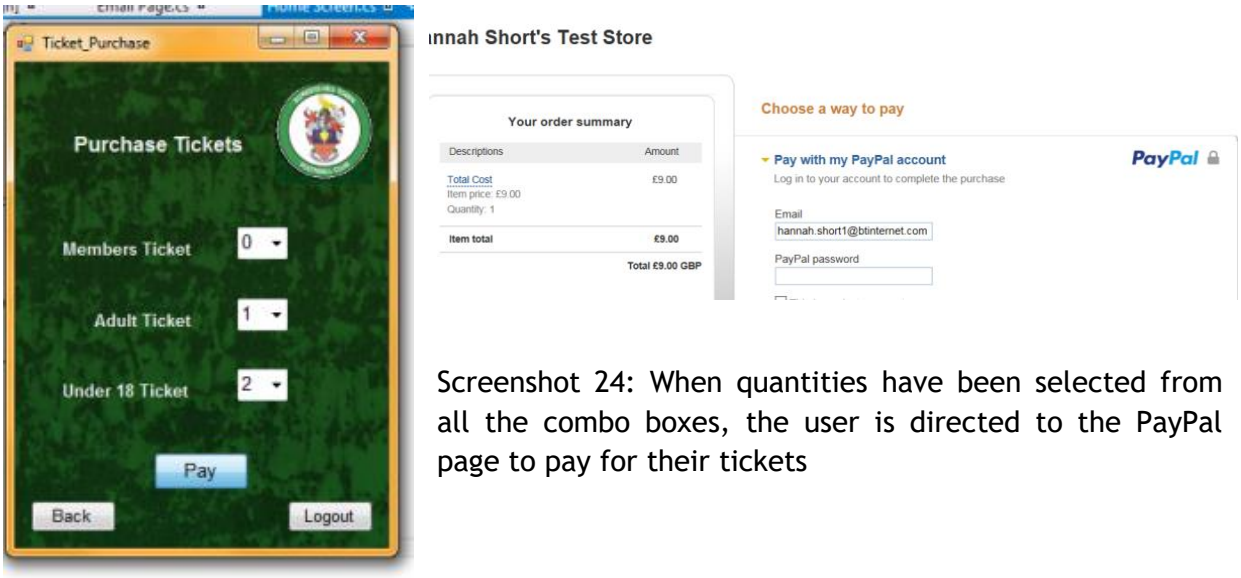
Screenshot 19: If the e-mail subject entered exceeds 40 characters, there is an error

Screenshot 20: Any team from the league can be selected from the combo box on the route form

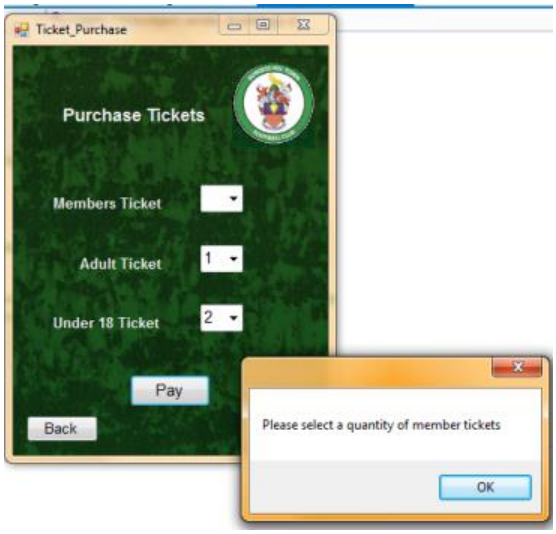
Screenshot 21: If no team has been selected from the options, there is an error

Screenshot 22: An option is selected from miles or kilometres

Screenshot 23: If neither the miles nor kilometres options have been selected, there is an error

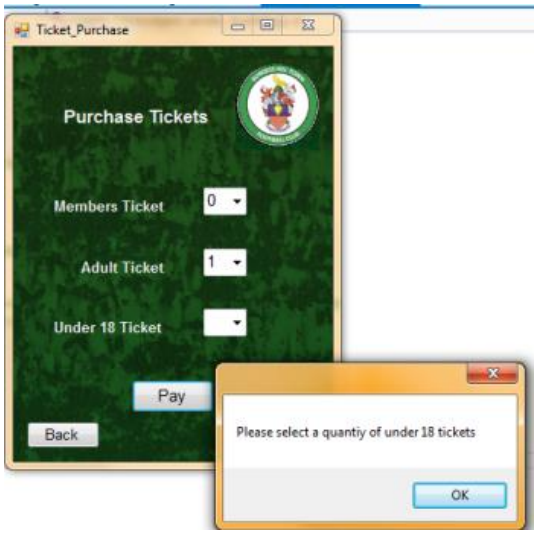


Screenshot 24: When quantities have been selected from all the combo boxes, the user is directed to the PayPal page to pay for their tickets



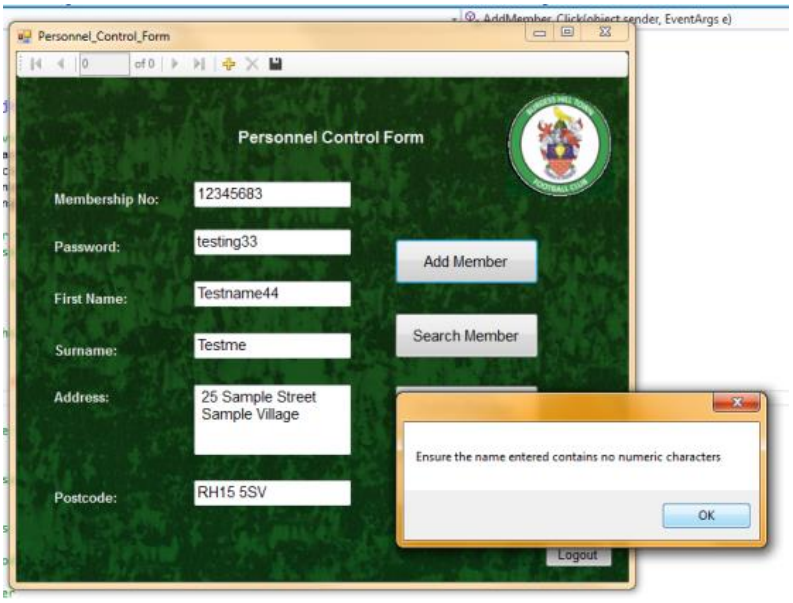
Screenshot 25: When nothing has been selected in the member's ticket menu, there is an error

Screenshot 26: When nothing has been selected in the adult ticket menu, there is an error

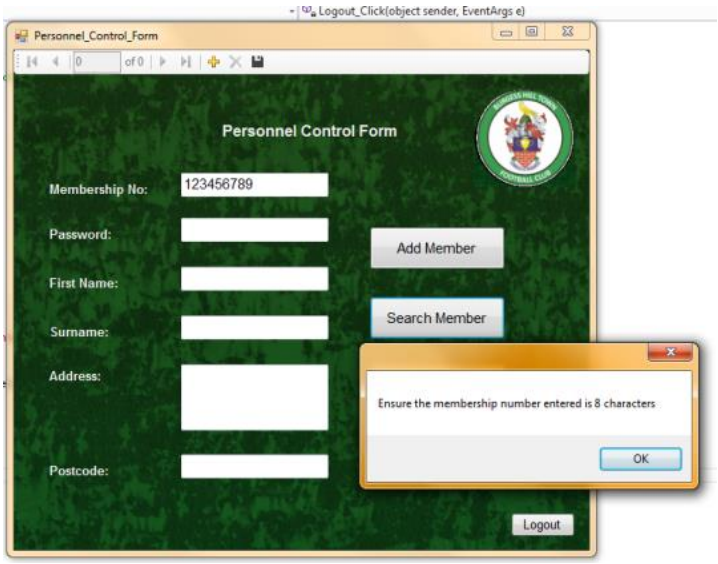


Screenshot 27: When nothing has been selected in the under 18 ticket menu, there is an error

Screenshot 28: The membership number that has been inputted into the membership number text box has been searched for because the membership number has been validated

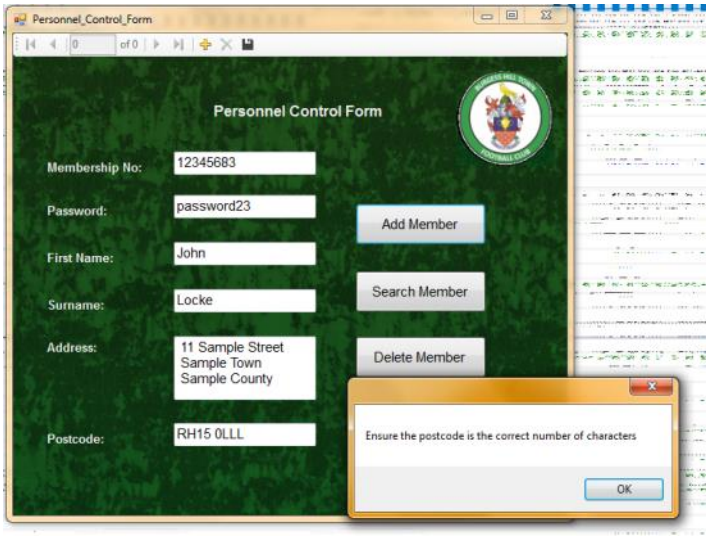


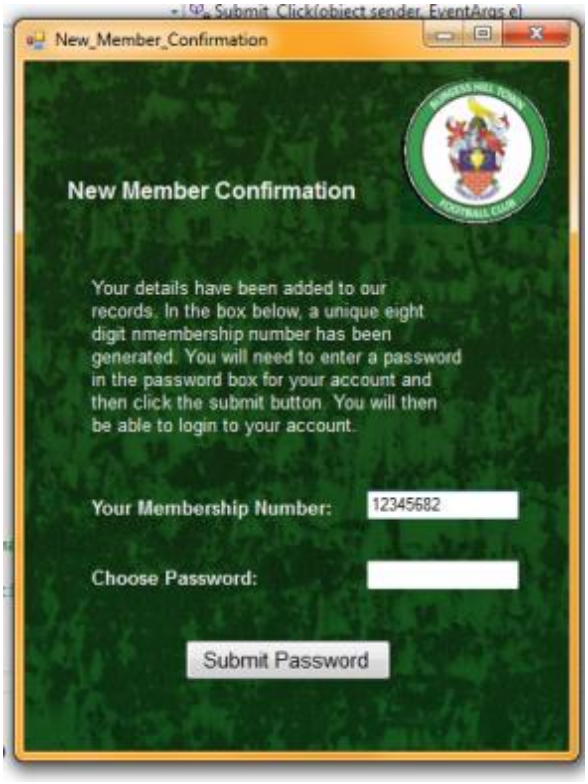
Screenshot 29: When a name is entered containing integers on the personnel control form, there is an error message



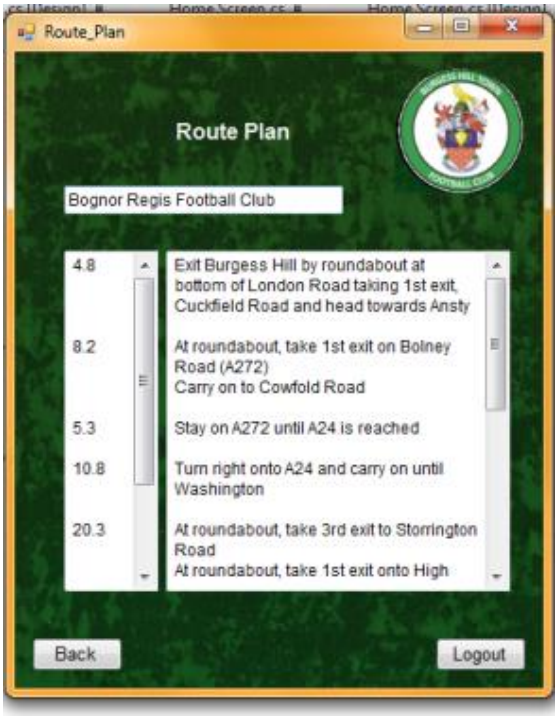
Screenshot 30: If a nine digit membership number is entered into the membership number box, there is an error message informing the staff member the number should be 8 digits

Screenshot 31: If a postcode is entered which is eight characters, there is an error message





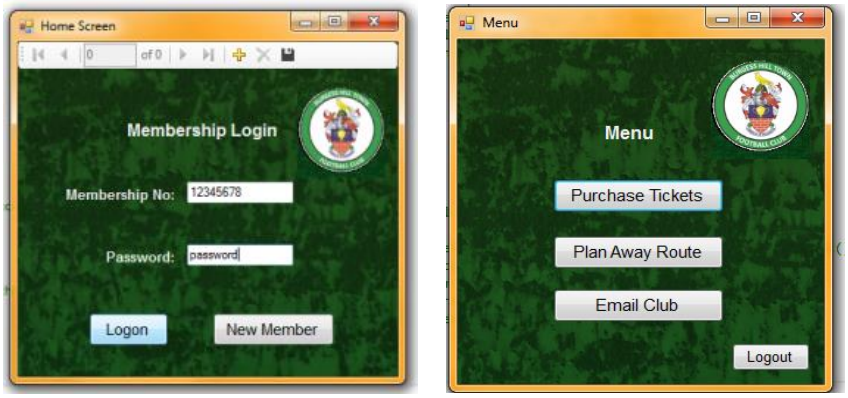
Screenshot 32: There is a number written in the membership number box of the member confirmation form



Screenshot 33: there is output in every text box of the route plan form

Algorithm Testing:

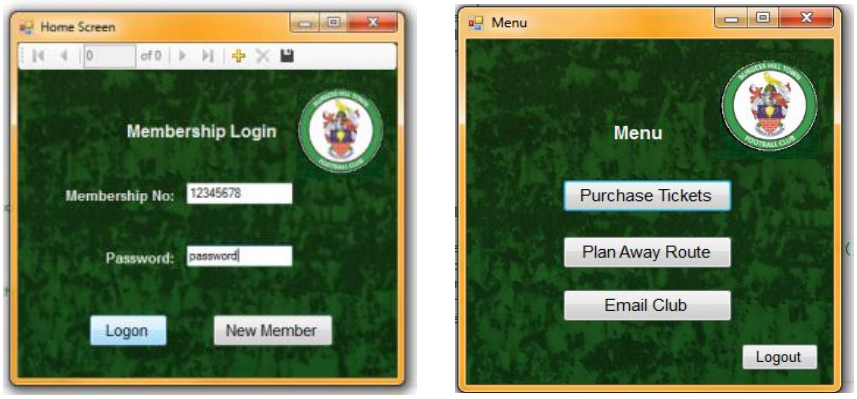
Screenshot 34: The menu form is opened when the membership number is validated by the validation algorithms in the home screen; this is proved the password is valid



Screenshot 35: If a membership number is entered which the system finds doesn't exist in the data set, there is an error

Screenshot 36: If the membership number entered by the number is one too many digits in length, an error message is displayed

Screenshot 37: If the password is present in the member's data set once the membership number has been validated, the menu is opened

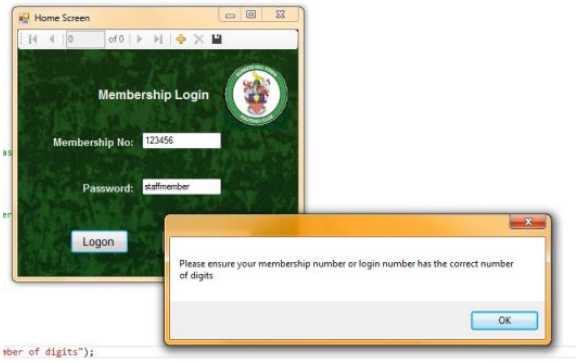


Screenshot 38: If the password entered doesn't exist, there is an error

Screenshot 39: If the login number is validated, provided there is also a valid password, the personnel control form will open

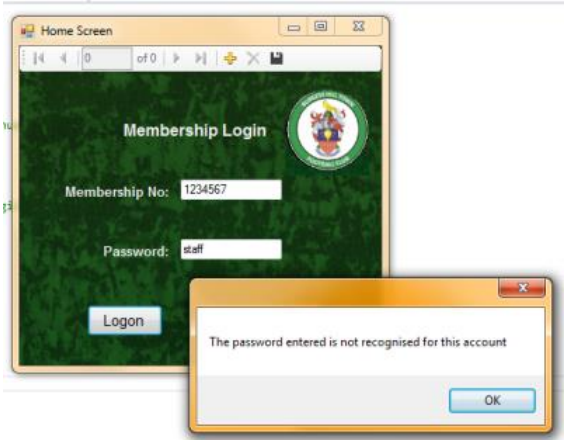
Screenshot 40: If a login number is entered which doesn't exist, there is an error

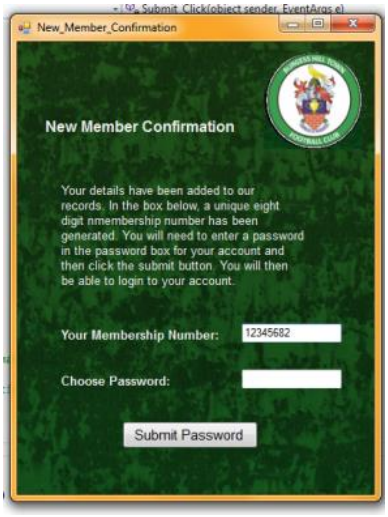
Screenshot 41: If the login number entered isn't enough characters, there is an error



Screenshot 42: If the password entered by the staff member is present in the personnel data set once the login number has been validated, the personnel control form is opened

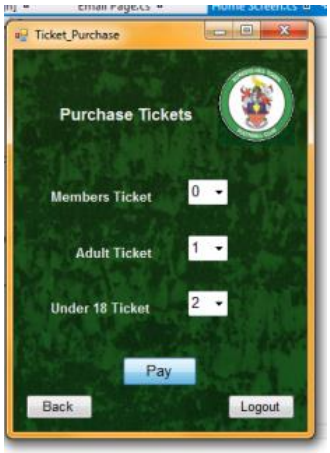
Screenshot 43: If the password entered doesn't exist, there is an error

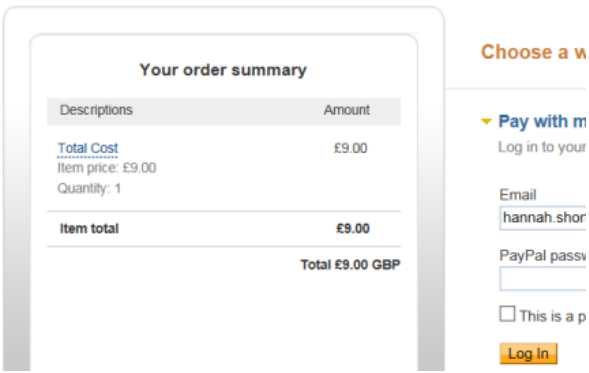




Screenshot 44: The number generated and written in the text box of the confirmation form is the next lowest possible number

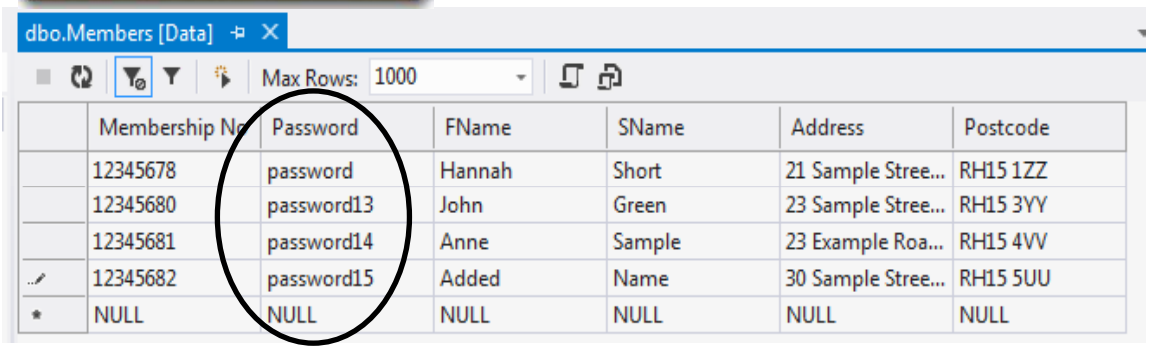
Screenshot 45: The cost calculated is the right amount outputted on the PayPal page





Screenshot 46: The club name, directions and distances are written to the boxes of the route plan form but the directions aren't relevant to the team

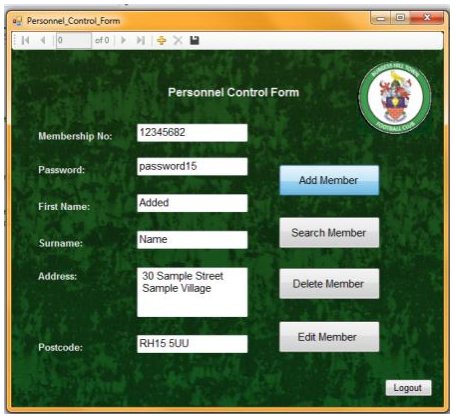
Screenshot 47: There is a method for encrypting the password but nothing is passed into it or returned as output



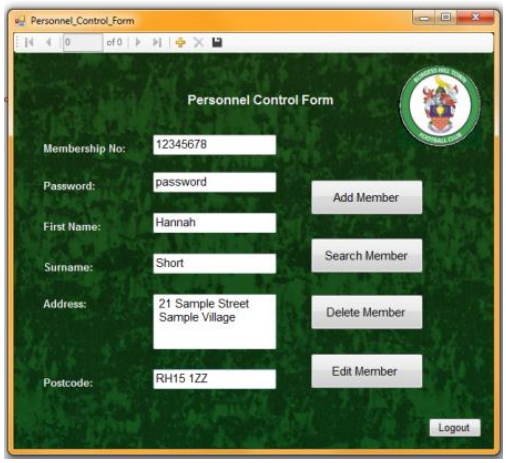
Functionality Testing:

Screenshot 48: Text entered on the home screen

Screenshot 49: Text entered on the personnel screen

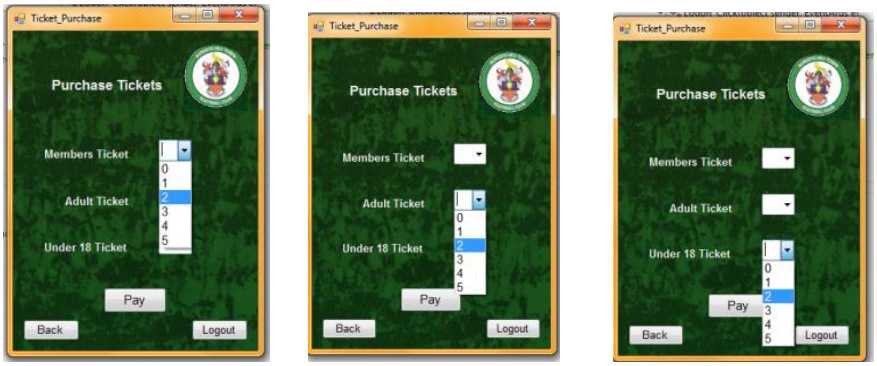


Screenshot 50: Text outputs to the personnel screen



Screenshot 51: Text entered on the new member screen

Screenshot 52: Combo boxes on the ticket purchase form function correctly

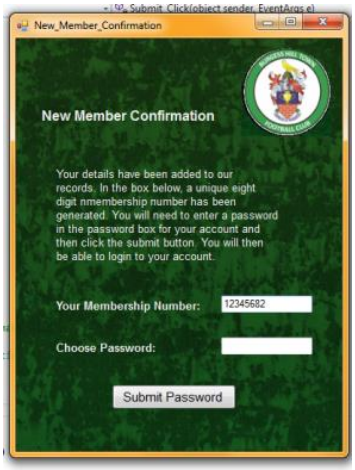


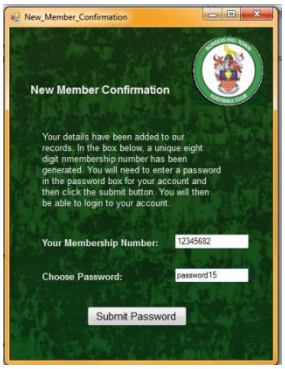
Screenshot 53: Combo box on the route form function

Screenshot 54: Text entered on the e-mail screen

Screenshot 55: The label on the member confirmation form outputs text when the form is loaded

Screenshot 56: Membership number is generated in text box when form loads





Screenshot 57: A password can be written in the text box on the confirmation screen

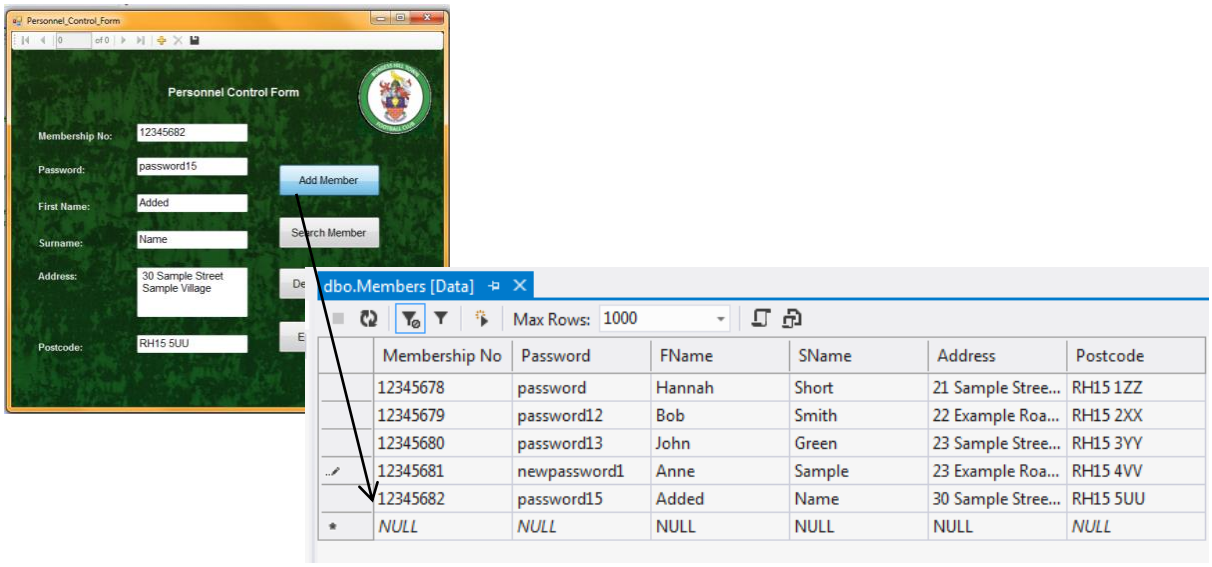
Screenshot 58: The label on the ticket confirmation form outputs text when the form is loaded

Screenshot 59: Text boxes on the route plan screen output text but not all text is visible

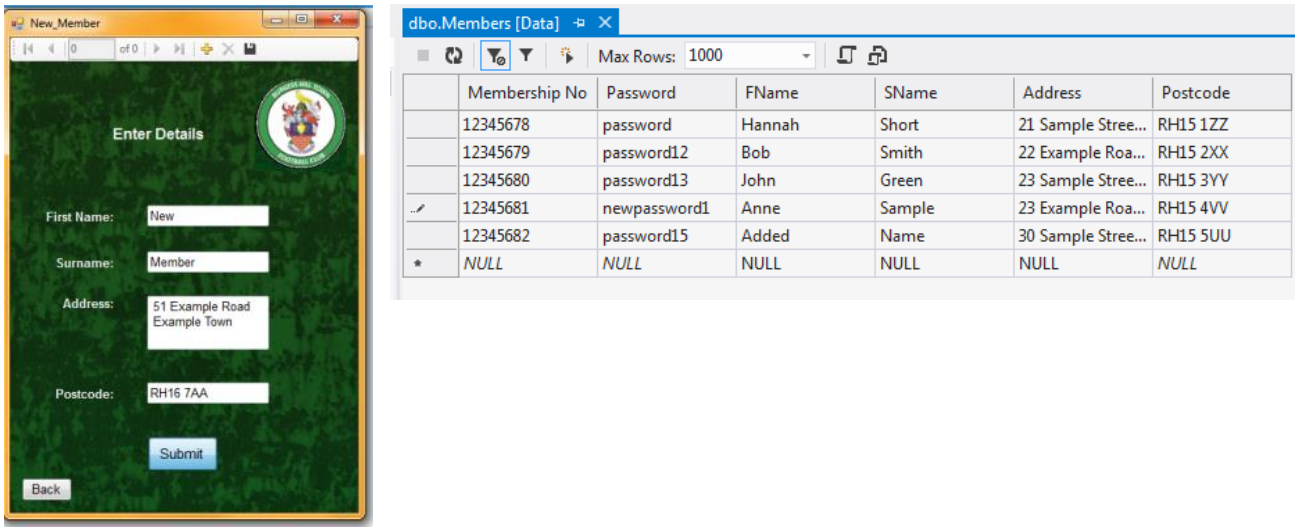
Screenshot 60: The label on the e-mail confirmation form outputs text when

Database Testing:

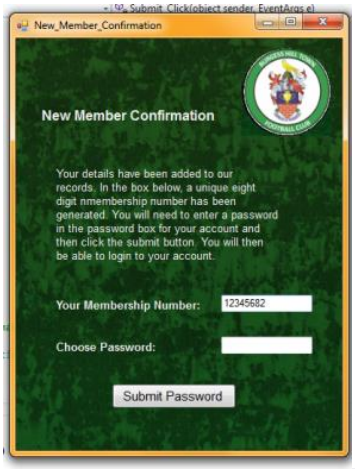
Screenshot 61: When a new member is added using the personnel control form, there is a new record in the members table

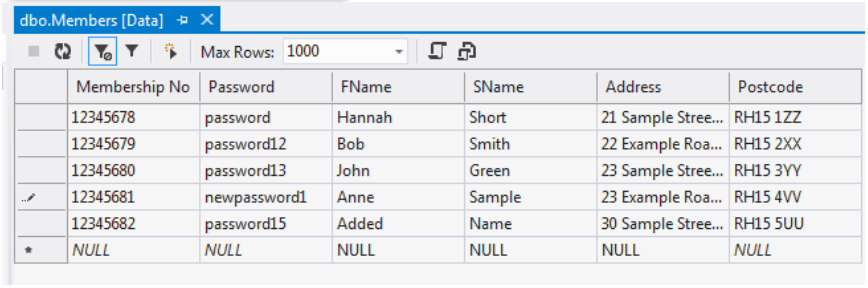


Screenshot 62: When the details from the new member form are submitted, no details are added and the table data remains the same because there is no key for the data to be added with it as the membership number isn't generated until the next form



Screenshot 63: The membership number can't be added after the other details because they are dependent on the number as the primary key of the table



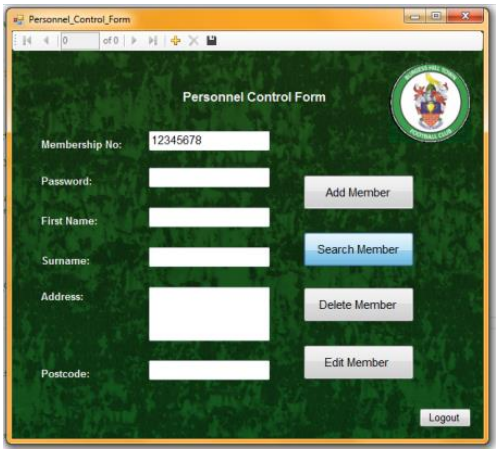


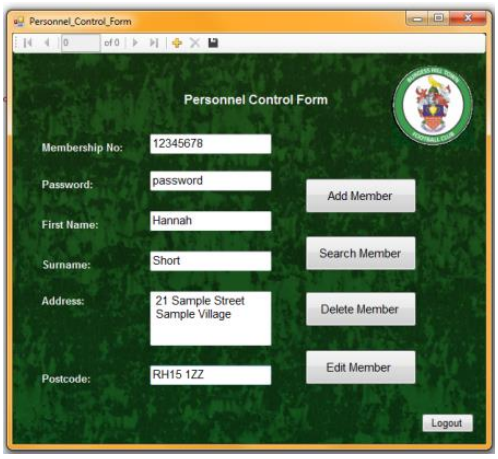
dbo.Members [Data]

Max Rows: 1000

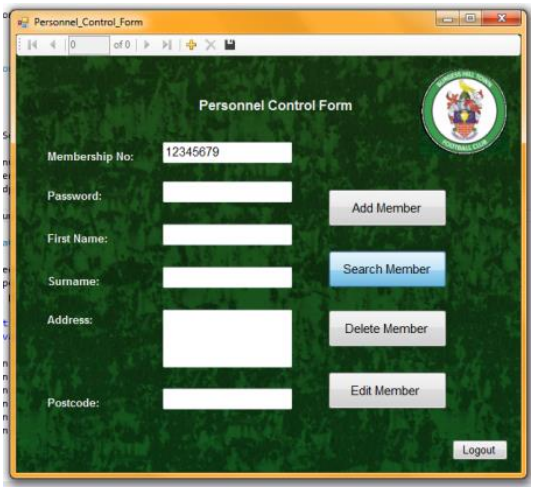
Membership No	Password	FName	SName	Address	Postcode
12345678	password	Hannah	Short	21 Sample Stree...	RH15 1ZZ
12345679	password12	Bob	Smith	22 Example Roa...	RH15 2XX
12345680	password13	John	Green	23 Sample Stree...	RH15 3YY
12345681	newpassword1	Anne	Sample	23 Example Roa...	RH15 4VV
12345682	password15	Added	Name	30 Sample Stree...	RH15 5UU
NULL	*NULL*	NULL	NULL	NULL	*NULL*

Screenshot 64: When the membership number of a member is entered and the search button is clicked, the details of the member are displayed in their respective text boxes





Screenshot 65: When a current member is deleted from the members list



dbo.Members [Data]

Max Rows: 1000

Membership No	Password	FName	SName	Address	Postcode
12345678	password	Hannah	Short	21 Sample Stree...	RH15 1ZZ
12345680	password13	John	Green	23 Sample Stree...	RH15 3YY
12345681	password14	Anne	Sample	23 Example Roa...	RH15 4VV
12345682	password15	Added	Name	30 Sample Stree...	RH15 5UU
NULL	NULL	NULL	NULL	NULL	NULL

Screenshot 66: When different member's details are written with their current membership number, their details are overwritten in the database

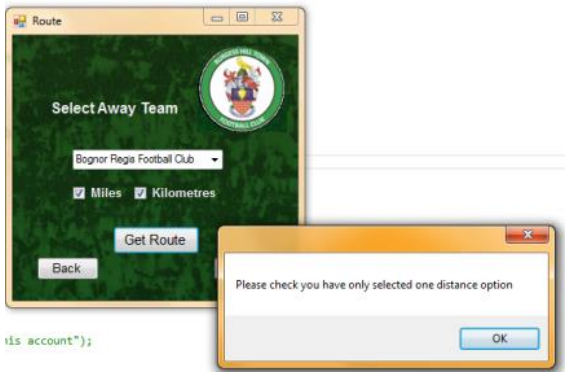
Modifying Failed Tests:

After testing the different aspects of my system, there were a couple of tests which failed to meet expectation therefore I will be trying to make further modifications to the system to ensure that the tests are successful

Input/output testing, test 6:

When both of the distance options are selected on the route form, there is no error. To modify this, I will add a bit of extra code validating the users' choices

```
if ((milesselected = true) && (kmselected == true))
                    {
                        MessageBox.Show("Please check you have only selected one
distance option");
                                                                    }
```

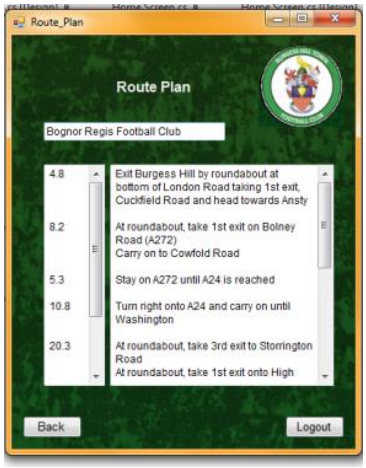


Algorithm testing, test 7:

Algorithm testing, test 8:

After testing the password encryption algorithm, I realised nothing was being passed into the function so I initialised the pass variable with the value that the user had inputted

```
private string encryptpass()
        {
            string pass = PasswordChoice.Text;
```



Functionality testing, test 12:

During the functionality testing I found that all the information on the route plan was not visible to the user. To rectify this, I added scrollbars to the two larger text boxes:

```
private void Direction_TextChanged(object sender, EventArgs e)
        {
            Direction.ScrollBars = ScrollBars.Vertical;
        }
        private void Distance_TextChanged(object sender,
EventArgs e)
        {
            Distance.ScrollBars = ScrollBars.Vertical;
                                        }
```

Database testing, test 2 & 3:

Currently a new member is unable to add themselves because the system attempts to add the membership number of the new member after the other details. I will change the form code so that the membership number is generated within the new member form and then the function is accessed by the new member confirmation so it can still be outputted to the screen

```
private void getnewnumber()
        {
            membnum = nm.generatemembnumber();
        }
```

dbo.Members [Data]

Max Rows: 1000

Membership No	Password	FName	SName	Address	Postcode
12345678	password	Hannah	Short	21 Sample Stree...	RH15 1ZZ
12345679	password12	Bob	Smith	22 Example Roa...	RH15 2XX
12345680	password13	John	Green	23 Sample Stree...	RH15 3YY
12345681	password14	Anne	Sample	23 Example Roa...	RH15 4VV
12345682	password15	Added	Name	30 Sample Stree...	RH15 5UU
12345683	password16	Extra	Person	40 Example Roa...	RH15 6TT
NULL	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

Algorithm Trace Tables:

Password Encryption Algorithm:

I will also test the password encryption using a trace table so each step of the algorithm can ber un through.

passLength	count	letter							
8		[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]

Route Planner Algorithm:

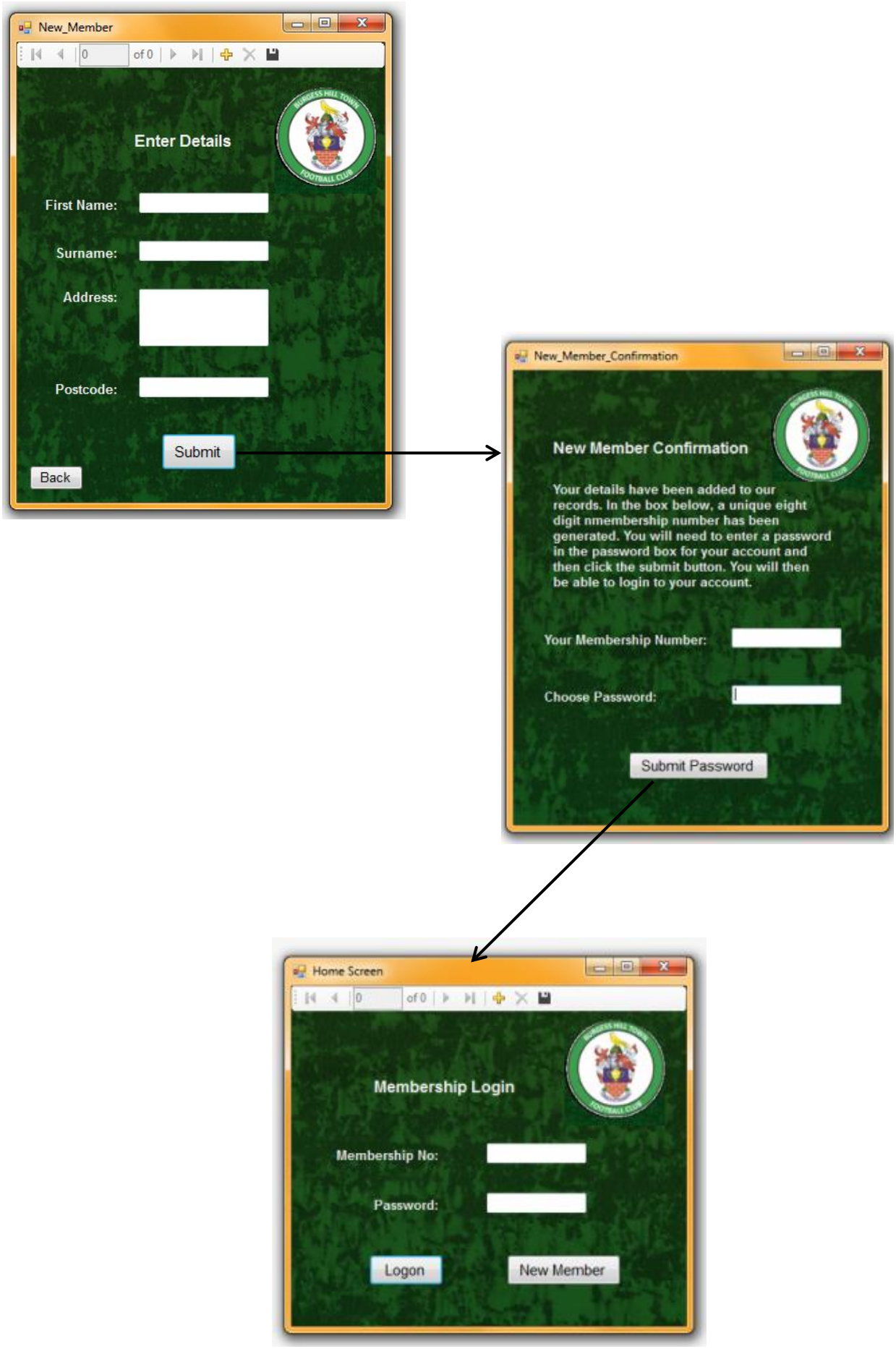
			Nodes Found								End Node Found											F
	Node	EndNode																				

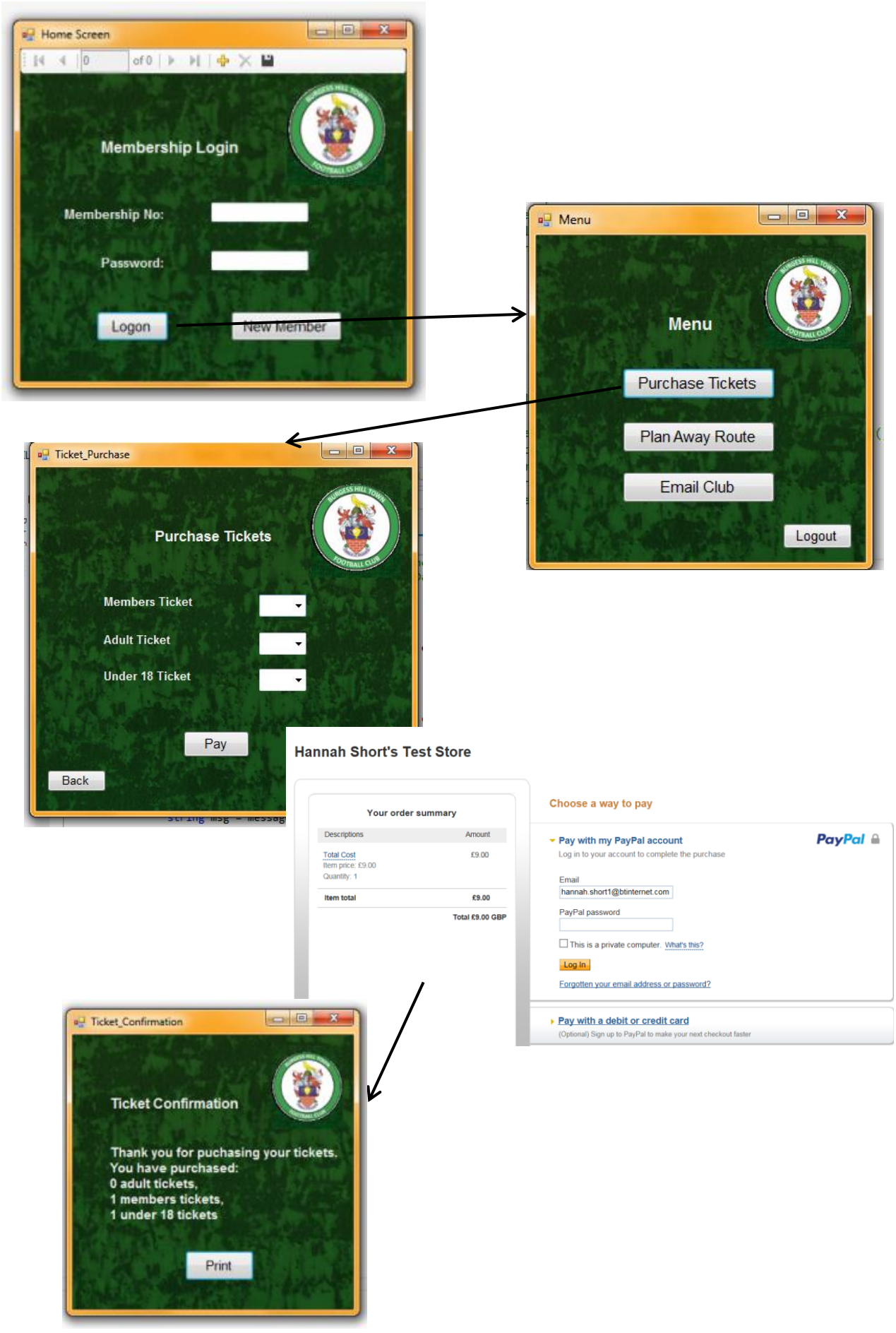
System Maintenance:

System Overview:

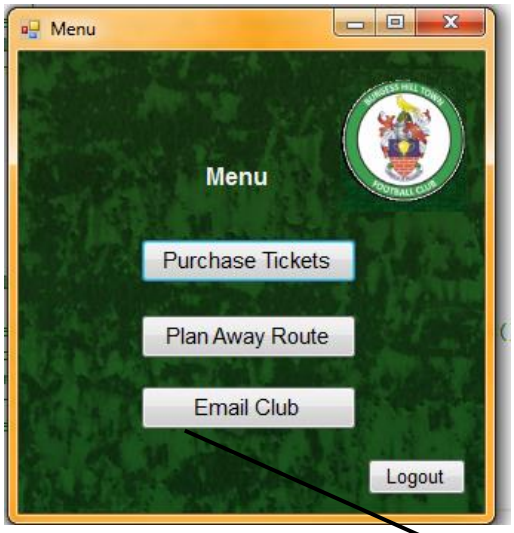
Structural Overview:

This is an overview of how all the forms in the system link together:


New_Member
0
of 0
Enter Details
First Name:
Surname:
Address:
Postcode:
Submit
Back
New_Member_Confirmation
New Member Confirmation
Your details have been added to our records. In the box below, a unique eight digit nmembership number has been generated. You will need to enter a password in the password box for your account and then click the submit button. You will then be able to login to your account.
Your Membership Number:
Choose Password:
Submit Password
Home Screen
0
of 0
Membership Login
Membership No:
Password:
Logon
New Member


Home Screen
Membership Login
Membership No:
Password:
Logon
New Member
Menu
Menu
Purchase Tickets
Plan Away Route
Email Club
Logout
Ticket_Purchase
Purchase Tickets
Members Ticket
Adult Ticket
Under 18 Ticket
Pay
Back
Hannah Short's Test Store
Your order summary
Descriptions
Amount
Total Cost
£9.00
Item price: £9.00
Quantity: 1
Item total
£9.00
Total £9.00 GBP
Choose a way to pay
Pay with my PayPal account
PayPal
Log in to your account to complete the purchase
Email
hannah.short1@btinternet.com
PayPal password
This is a private computer. What's this?
Log In
Forgotten your email address or password?
Pay with a debit or credit card
(Optional) Sign up to PayPal to make your next checkout faster
Ticket_Confirmation
Ticket Confirmation
Thank you for puchasing your tickets.
You have purchased:
0 adult tickets,
1 members tickets,
1 under 18 tickets
Print


Menu
Menu
Purchase Tickets
Plan Away Route
Email Club
Logout
Route
Select Away Team
Miles
Kilometres
Get Route
Back
Logout
Route_Plan
Route Plan
Bognor Regis Football Club
4.8
Exit Burgess Hill by roundabout at bottom of London Road taking 1st exit, Cuckfield Road and head towards Ansty
8.2
At roundabout, take 1st exit on Bolney Road (A272)
Carry on to Cowfold Road
5.3
Stay on A272 until A24 is reached
10.8
Turn right onto A24 and carry on until Washington
20.3
At roundabout, take 3rd exit to Storrington Road
At roundabout, take 1st exit onto High
Back
Logout


Menu
Menu
Purchase Tickets
Plan Away Route
Email Club
Logout


Email_Page
of 0
Email Message
First Name:
Surname:
Telephone:
Email Address:
Email Subject:
Email Message:
Back
Send
Logout


Emal_Confirmation
E-mail Confirmation
Thank you for your e-mail.
We will reply to you shortly
Logout


Home Screen
of 0
Membership Login
Membership No:
Password:
Logon
New Member
Personnel_Control_Form
of 0
Personnel Control Form
Membership No:
Password:
First Name:
Surname:
Address:
Postcode:
Add Member
Search Member
Delete Member
Edit Member
Logout

Detailed Algorithm Design:

These are some example of the most complex algorithms I have used in my program including an explanation of what the algorithm does, the algorithm written in pseudo code and the implemented code of the algorithm from my program.

1. Membership Number and Password Validation

   This algorithm checks to see whether the membership number and the password that the member has entered on the home screen are valid by checking that they exist in the members table of the database and contain the right number of digits.

   Algorithm in Pseudo Code:

```
if MembershipNo = null
  PRINT "Please enter your membership number"
else
if Length(MembershipNo) <> 8
  PRINT "Ensure your membership number is 8 digits"
else
forEach digit in number
  If digit <> numeric
    PRINT "Ensure your membership number contains only numeric
characters"
else
  SELECT MembershipNo
  FROM Members
       WHERE MembershipNo = MembershipNoTextBox.Text
  if forEach
        Members.MembershipNo <> MembershipNoTextBox.Text
          PRINT "There is no record of this membership number"
end
if Password = null
  PRINT "Please enter your password"
else
if length(Password) < 8 or length(Password) > 20
  PRINT "Your password needs to between 8 and 20 characters"
else
if
  SELECT Password
  FROM Members
    WHERE Members.MembershipNo = MemberhipNoTextBox.Text
    AND Members.Password = PasswordTextBox.Text
== false
  PRINT "Invalid Password"

end
```

Program Code for Algorithm:

```
private bool acceptmno()
            {

                int membnum = Convert.ToInt32(membership_NoTextBox.Text);

                memberDataContext context = new memberDataContext();

                var peopledata =
                    from person in context.Members
                    where person.Membership_No == membnum
                    select new { person.Membership_No };
                foreach (var person in context.Members)
                {
                    person.Membership_No = membnum;
                }
                return true;
            }


private bool acceptmpass()
            {
                memberDataContext context = new memberDataContext();
                var peopledata =
                    from person in context.Members
                    where person.Password == passwordTextBox.Text
                    select new { person.Password };
                foreach (var person in context.Members)
                {
                    person.Password = passwordTextBox.Text;
                }
                return true;
            }


if (membership_NoTextBox.Text == " ")
                {
                    MessageBox.Show("Please enter your membership number");
                }
                else
                    if (passwordTextBox.Text == " ")
                    {
                        MessageBox.Show("Please enter your password");
                    }
                    else
                        if ((membership_NoTextBox.Text.Length) > 8)
                        {
                            MessageBox.Show("Please ensure your membership number or
login number has the correct number of digits");
                        }
                        else
                            if ((membership_NoTextBox.Text.Length) < 7)
                            {
                                MessageBox.Show("Please ensure your membership number
or login number has the correct number of digits");
                            }
```

2. Login Number and Password Validation

   This algorithm ensures that the login number and the password that have been inputted into the text boxes on the home screen exist in the personnel table of the database and contain the right number of digits for a valid staff login number.

   Algorithm in Pseudo Code:

```
if LoginNo = null
  PRINT "Please enter your login number"
else
if Length(MembershipNo) <> 7
  PRINT "Ensure your login number is 7 digits"
else
forEach digit in number
  If digit <> numeric
    PRINT "Ensure your login number contains only numeric
characters"
else
  SELECT LoginNo
  FROM Personnel
       WHERE LoginNo = MembershipNoTextBox.Text
  if forEach
        Personnel.LoginNo <> MembershipNoTextBox.Text
          PRINT "There is no record of this login number"
end
if Password = null
  PRINT "Please enter your password"
else
if length(Password) < 8 or length(Password) > 20
  PRINT "Your password needs to between 8 and 20 characters"
else
if
  SELECT Password
  FROM Members
    WHERE Members.MembershipNo = MemberhipNoTextBox.Text
    AND Members.Password = PasswordTextBox.Text
== false
  PRINT "Invalid Password"
end
```

Program Code for Algorithm:

```
private bool acceptplogin()
        {
            int loginno = Convert.ToInt32(membership_NoTextBox.Text);

            PersonnelDataContext context = new PersonnelDataContext();
            var peopledata =
                from person in context.Personnels
                where person.Login_No == loginno
                select new { person.Login_No };
            foreach (var person in context.Personnels)
            {
                person.Login_No = loginno;
            }

            return true;

        }

       {
           PersonnelDataContext context = new PersonnelDataContext();
           var peopledata =
               from person in context.Personnels
               where person.Password == passwordTextBox.Text
               select new { person.Password };
           foreach (var person in context.Personnels)
           {
               person.Password = passwordTextBox.Text;
           }

           return true;
       }


if (membership_NoTextBox.Text == " ")
                {
                    MessageBox.Show("Please enter your membership number");
                }
                else
                    //Checks there is data present in the password text box
                    if (passwordTextBox.Text == " ")
                    {
                        MessageBox.Show("Please enter your password");
                    }
                    else
                        //Checks that the number entered is the correct length
                        if ((membership_NoTextBox.Text.Length) > 8)
                        {
                            MessageBox.Show("Please ensure your membership number or
login number has the correct number of digits");
                        }
                        else
                            if ((membership_NoTextBox.Text.Length) < 7)
                            {
                                MessageBox.Show("Please ensure your membership number
or login number has the correct number of digits");
                            }
```

3. Add Member to Database

A member can be added to the database in either the new member form or when the add button is clicked in the personnel control form. When a member adds them self in the new member form, the personal details they have entered are validated and then added to the database.

Algorithm in Pseudo Code:

```
INSERT INTO Members (MembershipNo, Password, FName, Surname,
Address, Postcode)
MEMBER     (MembershipNoTextBox.Text,     PasswordTextBox.Text,
FNameTextBox.Text,  SurnameTextBox.Text,  AddressTextBox.Text,
PostcodeTextBox.Text)
```

Program Code for Algorithm:

```
private void AddMember_Click(object sender, EventArgs e)
        {
            memberDataContext context = new memberDataContext();

            Member m = new Member
            {
                FName = fName.Text,
                SName = sName.Text,
                Address = address.Text,
                Postcode = postcode.Text
            };

            context.Members.InsertOnSubmit(m);

            try
            {
                context.SubmitChanges();
            }
            catch (Exception e1)
            {
                Console.WriteLine(e1);
                context.SubmitChanges();
            }
        }
```

4. Search for Member in Database

A member of staff is able to search for a particular club member in the personnel control form, if a data item they have entered into one of the text boxes matches the data for that member in the database, the other data items relating to that member will be displayed in the other boxes on the form.

Algorithm in Pseudo Code:

```
SELECT  MembershipNo
FROM Members
WHERE MembershipNo = MembershipNoTextBox.Text

SELECT  Password
FROM Members
WHERE Password = PasswordTextBox.Text

SELECT  FName
FROM Members
WHERE FName = FNameTextBox.Text

SELECT  Surname
FROM Members
WHERE Surname = SurnameTextBox.Text

SELECT  Address
FROM Members
WHERE Address = AddressTextBox.Text

SELECT  Postcode
FROM Members
WHERE Postcode = PostcodeTextBox.Text
```

Program Code for Algorithm:

```
private void SearchMember_Click(object sender, EventArgs e)
        {
            bool membnumberlength = checkmembnumber();
            bool presentnumeric = checknumericinname();
            bool validpc = validatepclength();

            int membnum = Convert.ToInt32(MembNumber.Text);

            memberDataContext context = new memberDataContext();

            var peopledata =
                from person in context.Members
                where person.Membership_No == membnum

                select person;
```

```
            foreach (var person in peopledata)
            {
                person.Membership_No = membnum;
                person.Password = password.Text;
                person.FName = fName.Text;
                person.SName = sName.Text;
                person.Address = address.Text;
                person.Postcode = postcode.Text;
            }


                  }
```

5. Delete Member in Database

   If a member wants to cancel their membership, they can be deleted by a staff member clicking the delete member in the personnel control form. All their details will be removed from the database.

   Algorithm in Pseudo Code:

```
DELETE FROM Members
WHERE MembershipNo = MembershipNoTextBox.Text
```

   Program Code for Algorithm:

```
private void DeleteMember_Click(object sender, EventArgs e)
        {
            int membnum = Convert.ToInt32(MembNumber.Text);

            memberDataContext context = new memberDataContext();

            var peopledata =
                from person in context.Members
                where person.Membership_No == membnum
                select person;

                foreach (var member in peopledata)
            {
                context.Members.DeleteOnSubmit(member);
            }

            try
            {
                context.SubmitChanges();
            }
            catch (Exception e1)
            {
                Console.WriteLine(e1);
            }
        }
```

6. Edit Members Details in Database

When a member needs to change one of their personal details, the club personnel are able to change the data recorded in the database by entering all the personal details of the member into the text boxes on the form and then clicking edit member. The algorithm will search for the member in the database with the membership number entered and then all the other items for that member will be overwritten with the data in the text boxes

Algorithm in Pseudo Code:

```
UPDATE Member
SET Password = PasswordTextBox.Text, FName =
FNameTextBox.Text, Surname = SurnameTextBox.Text, Address =
AddressTextBox.Text, Postcode = PostcodeTextBox.Text
WHERE MembershipNo = MembershipNoTextBox.Text
```

Program Code for Algorithm:

```
private void EditMember_Click(object sender, EventArgs e)
        {
            bool membnumberlength = checkmembnumber();
            bool presentnumeric = checknumericinname();
            bool validpc = validatepclength();

            int membnum = Convert.ToInt32(MembNumber.Text);

            memberDataContext context = new memberDataContext();

            var editpeople =
                from person in context.Members
                where person.Membership_No == membnum
                select person;
                foreach (Member person in editpeople)
                {
                    person.Membership_No = membnum;
                    person.Password = password.Text;
                    person.FName = fName.Text;
                    person.SName = sName.Text;
                    person.Address = address.Text;
                    person.Postcode = postcode.Text;
                }

                try
                {
                    context.SubmitChanges();
                }
                catch (Exception e1)
                {
                    Console.WriteLine(e1);
                }

                  }
```

7. Generate membership Number for new member

   When a new member is added to the system, they need their own unique membership number. Once their details have been validated and submitted, a membership number will be generated when the new member confirmation form loads. An algorithm will be run where the next consecutive eight digit membership number that isn't already stored in the members' table is given to the new member and stored as their own number

   Algorithm in Pseudo Code

```
GenerateNumber
 NewMembNumber = Max(Members.MembershipNo) + 1
endGenerateNumber
```

   Program Code for Algorithm:

```
private int generatemembnumber()
        {
            int maxnumber = 0;
            int newnumb;

            memberDataContext context = new memberDataContext();

            var peopledata =
                from person in context.Members
                where person.Membership_No == maxnumber
                select new { person.Membership_No };

            foreach (var person in context.Members)
            {
                if (person.Membership_No > maxnumber)
                {
                    person.Membership_No = maxnumber;
                }
            }
            newnumb = maxnumber + 1;
            return newnumb;
                  }
```

8. Calculate Ticket Price and pay for tickets

   The total cost a member will have to pay in a single transaction will be calculated by multiplying the price of each ticket type by the number of tickets of that type being purchased, these totals will then be added together to get the total cost of all the tickets being bought.

   Algorithm in Pseudo Code:

```
CalcTotalCost(AdultTicketQuantity, MemberTicketQuantity)
 AdultsCost = AdultTicketQuantity * 9
 MembersCost = MemberTicketQuantity * 5
 TotalCost = AdultsCost + MembersCost
EndCalcTotalCost
```

   Program Code for Algorithm:

```
private int CalcCost()
        {
            int totalcost;
            totalcost = ((adultquantity * 9) + (membersquantity * 5));
            return totalcost;
                  }
```

9. Route Planner Code

A traversal algorithm will be run when the user clicks the get route button on the route form to find the edges between the Burgess Hill Football club node and the destination node the member is travelling to

Algorithm in Pseudo Code:

```
GetRoute (Node, EndNode)
Found[Node] ← true
   if Node = EndNode then Reached ← true
   for each NextNode N linked to Node
   if Found[N] = false then GetRoute(Node,EndNode)
   end for
   DestinationReached[Node] ← true
   Output found nodes
End GetRoute
```

Program Code for Algorithm:

```
private bool calcroute(int Node, int EndNode)
        {
            int[,] weights = new int[,]
            {
            {0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,1,0,0,0},
            {0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0},
            {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0},
            {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0},
            {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0},
            {0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
            {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0},
            {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0},
            {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},
            {1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0},
            {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0},
            {0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,1,0,0,0},
            {0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0},
            {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0},
            {0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,1,0,0,0},
            {0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0},
            {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0},
            };

            bool[] Found = new bool[30];
            bool[] DestinationReached = new bool[30];
            bool Reached = false;

            if (Node == EndNode)
            {
                Reached = true;
            }
            for (int i = 0; i < 30; i++)
```

```
{
    if (Found[i] == false)
    {
        calcroute(Node, EndNode);
    }
}
DestinationReached[Node] = true;

return DestinationReached[Node];
      }
```

10. Encrypt Password Code

For system security, the members' passwords are stored in an encrypted form in the database. When they log on, the password they have entered in the password box is encrypted before being checked against the encrypted passwords in the database. A substitution cipher is used to encrypt the passwords. Also when a new member chooses the password for their account, it will be encrypted before being added to the members' database

Algorithm in Pseudo Code:

```
EncryptPassword
 count ← 0
 do until count = password length
   count = count + 1
   get character at position count
   letter[count] = ASCII(letter[count])
   letter[count] = letter[count] + 4
   if 122 < letter[count] < 127 then letter[count] = 97 +
(letter[count] – 123)
 loop
 count ← 0
 do until count = password length
   count = count + 1
   (letter[count]) = CHR(letter[count])
 loop
end EncryptPassword
```

Program Code for Algorithm:

```
private string encryptpass()
        {
            string pass = PasswordChoice.Text;
            char [] letter = new char[10];
            int[] x = new int[10];

            for (int i = 0; i <= pass.Length; i++)
            {

                letter[i] = Convert.ToChar(pass.Substring(i, i + 1));
                x[i] = (int)letter[i];
                x[i] = x[i] + 4;
                if ((x[i] > 122) && (x[i] < 127))
                {
                    x[i] = 97 + (x[i] - 123);

                }
            }

            return pass;}
```

Procedure and Variable Lists:

Procedures:

This is a list of the procedures and functions that I have used in my code. It describes the type of access they have, their name, a short description of what their purpose is, the parameters they take as well as what outputs they give to the user or the values they return.

Access	Name	Description	Outputs/Returned Values
Private	acceptmno	Checks to see whether the number the member has entered exists in the members table in the database	Boolean – returns true if membership number is valid
Private	acceptmpass	Checks to see whether the password the member has entered corresponds to the membership number entered	Boolean – returns true if the password is valid
Private	acceptplogin	Checks to see whether the number the member of staff has entered exists in the personnel table in the database	Boolean – returns true if the login number is valid
Private	acceptppass	Checks to see whether the password the member of staff has entered corresponds to the login no entered	Boolean – returns true if the password entered is valid
Private	encryptpass	Uses a substitution cipher to encrypt the password entered by the user	String – returns the encrypted form of the password
Private	Logon_Click	When the user clicks the logon button, data is checked to be valid	Void (Opens either menu form or personnel control form if data is valid, outputs error message otherwise)
Private	NewMember_Click	When the user clicks the new member button on the home screen	Void (Opens new member form)
Private	PurchaseTickets_Click	When the user clicks the purchase tickets option on the menu form	Void (Opens the ticket purchase form)
Private	PlanAwayRoute_Click	When the user clicks the plan route button on the menu form	Void (Opens the route form)
Private	EmailClub_Click	When the user clicks the contact club button on the menu form	Void (Opens the email form)
Private	Logout_Click	Directs the user back to the home screen regardless of where they are	Void (Opens the home screen form)

Access	Name	Description	Outputs/Returned Values
Private	checkmemberselected	Checks to see if a quantity has been selected from the members ticket menu on the purchase tickets form	Boolean – If the quantity of member tickets has been selected, true is returned
Private	checkadultselected	Checks to see if a quantity has been selected from the adult ticket menu on the purchase tickets form	Boolean – If the quantity of adult tickets has been selected, true is returned
Private	checku18selected	Checks to see if a quantity has been selected from the under 18 ticket menu on the purchase tickets form	Boolean – If the quantity of under 18 tickets has been selected, true is returned
Public	getmemberquantity	Gets the number of member tickets that have been selected by the user	Integer – Returns the member ticket quantity
Public	getadultquantity	Gets the number of adult tickets that have been selected by the user	Integer – Returns the adult ticket quantity
Public	getu18quantity	Gets the number of under 18 tickets that have been selected by the user	Integer – Returns the under 18 ticket quantity
Public	CalcCost	Calculates the total amount of the user's purchase	Integer – Returns the calculated value of the purchase
Private	writepaypalhtml	Writes a html file using the return value from CalcCost	Void
Private	Pay_Click	When the user clicks the pay button on the purchase tickets form	Void (Opens PayPal page if options have all been selected otherwise there is an error)
Private	Back_Click	When the user clicks the back button on a form	Void (Opens previous form)
Public	getselectedteam	Gets the name of the team the user has selected from the combo box on the route form	String – returns the name of the team
Private	getnode	Gets the corresponding node on the graph of the team the user has chosen	Integer – returns the index of the node
Private	calcroute	Calculate the edges of the graph that will be traversed to get to the node of the team the user has selected	Boolean – Returns true if the destination node has been reached
Public	MilesSelected	Checks if the Miles checkbox has been selected on the route form	Boolean – Returns true of the checkbox has been checked

Access	Name	Description	Outputs/Returned Values
Public	KMSelected	Checks if the Kilometres checkbox has been selected on the route form	Boolean – Returns true if the checkbox has been checked
Private	GetRoute_Click	When the user clicks the get route button on the route form	Void (Opens the route plan if the necessary options have been selected, otherwise an error message is shown)
Private	checkdatapresent	Checks if data has been entered into all of the fields on the form	Boolean – Returns true if there is data in all of the text boxes
Private	checkalphanumber	Checks if there is a presence of a letter when a telephone number is entered by the user	Boolean – Returns true if there are any letters in the number
Private	checksubjectlength	Checks the length of the e-mail subject entered on the e-mail form	Boolean – Returns true if the subject is less than or equal to 40 characters, the accepted length
Private	checkmessagelength	Checks the length of the e-mail message entered on the e-mail form	Boolean – Returns true if the message is less than or equal to 400 characters, the accepted length
Private	Send_Click	When the send button is clicked on the e-mail form	Void (Opens the e-mail confirmation form if all details are valid, otherwise an error message is displayed depending on the error)
Public	getticketquantities	Gets the ticket quantities selected on the previous form	Void
Private	getteam	Gets the team name of the club selected on the previous form	Void
Private	checkmembnumber	Checks if the length of the membership number entered by the user is valid	Boolean – Returns true if the membership number is the correct length
Private	checkalphainname	Checks there are no alpha characters in the membership number entered by the user	Boolean – Returns true if there is an alpha character in the number
Private	checknumericinname	Checks that first names and surnames entered by the user don't contain numeric characters	Boolean – Returns true if there is a numeric characters in a name
Private	validatepclength	Checks the postcode entered by the user is a valid length	Boolean – Returns true if the postcode is a valid length

Access	Name	Description	Outputs/Returned Values
Private	AddMember_Click	When the user clicks the add member button on the personnel control form	Void (Adds the member to the member's table of the database if all the details are valid)
Private	DeleteMember_Click	When the user clicks the delete member button on the personnel control form	Void (Deletes the member from the member's table of the database if all the details are valid)
Private	EditMember_Click	When the user clicks the edit member button on the personnel control form	Void (Updates the details of a member from the table in the database)
Private	SearchMember_Click	When the user clicks on the search member button on the personnel control form	Void (Displays the details of the membership number entered by the user)
Private	generatemembnumber	Generates the lowest possible membership number for a new member by finding the maximum number in the table and incrementing it by one	Integer – Returns the generated membership number
Private	Submit_Click	When the submit button is clicked on the new member form	Void (If all the details entered by the user are valid, they receive a new member confirmation from the system)

Variables:

This is a list of all the variables I have used in my program code. I have given the names, data types and short descriptions of their purpose.

Name	Type	Description
Membership Number	int	The eight digit membership number unique to every club member, stored in the member's table of the database
Password	string	Every club member and staff member has a password of their choice to login in to their account, they are stored in the members and personnel table
Login Number	int	The seven digit number unique to every staff member using the system, stored in the personnel table of the database
First Name	string	First names of the club members, stored in the member's table of the database
Surname	string	Surnames of the club members, stored in the member's table of the database
Address	string	Addresses of the club members, stored in the member's table of the database
Postcode	string	Postcodes of the club members, stored in the member's table of the database
membnum	Int	Takes the value of the converted string entered into the text box for membership number
loginno	int	Takes the value of the converted string entered into the text box for membership number
acptmembnum	bool	Takes the returned value of the acceptmno function
acptmempass	bool	Takes the returned value of the acceptmpass function
acptstafflog	bool	Takes the returned value of the acceptplogin function
acptstaffpass	bool	Takes the returned value of the acceptppass function
pass	string	Takes the value of the string entered in the password text box and gets returned as the encrypted form
letter	char[]	An array containing all the letters of the string pass
x	int[]	An array containing all the ascii values of the characters of the string pass
membnum	int	Takes the converted integer value of the membership number text box
loginno	int	Takes the converted integer value of the password text box
selected	bool	True if the selected index of the combination box takes a value of more than negative one

Name	Type	Description
quantity	int	Takes the value of the number selected by the user from the ticket category menus
adultquantity	int	Takes the returned value of the getadultquantity function
membersquantity	int	Takes the returned value of the getmemberquantity function
totalcost	int	The result of the calculation multiplying the ticket quantities by their respective prices
htmllines	string[]	An array containing all the lines waiting to be written to the html file 'Pay'
memberselect	bool	Takes the returned value of the checkmemberselected function
adultselect	bool	Takes the returned value of the checkadultselected function
u18select	bool	Takes the returned value of the checku18selected function
club	string	The name of the club selected by the user on the route form
team	string	Takes the returned value of the getselectedteam function
node	int	Takes the index of the node on the graph of the team selected
weights	int[,]	A 2d array representing an adjacency matrix of the youth team league graph
Found	bool[]	An array storing Boolean values of which nodes have been discovered on the graph
DestinationReached	bool[]	An array with a value of true if the destination node has been reached on the graph
Reached	bool	Takes a value of true if the parameter Node passed into the function is the same as the destination node
check	bool	Takes of value of true if a checkbox has been selected
milesselected	bool	Takes the returned value of the MilesSelected function
kmselected	bool	Takes the returned value of the KMSelected function
presentdata	bool	Takes a value of true if there is data in all the text boxes of a form
IsAlpha	bool	Takes a value of true if a number contains an alpha character
c	char	The digits in a number. Not an array variable because no values need to be stored
length	bool	If the length of the email subject entered by the user is valid, the variable takes a value of true
mlength	bool	If the length of the email message entered by the user is valid, the variable takes a value of true

Name	Type	Description
datapresent	bool	Takes the returned value of the checkdatapresent function
presentalpha	bool	Takes the returned value of the checkalphanumber function
validsubjectlength	bool	Takes the returned value of the checksubjectlength function
validmessagelength	bool	Takes the returned value of the checkmessagelength function
mailaddress	string	Takes the value of the string in the address text box of the e-mail form
subject	string	Takes the value of the string in the subject text box of the e-mail form
msg	string	Takes the value of the string in the message text box of the e-mail form
adultticket	int	Takes the returned value of the getadultquantity function from the previous form
memberticket	int	Takes the returned value of the getmemberquantity function from the previous form
u18ticket	int	Takes the returned value of the getu18quantity function from the previous form
mileschecked	bool	Takes the returned value of the MilesSelected function on the previous form
kmchecked	bool	Takes the returned value of the KMSelected function on the previous form
distance	string	Takes the value of the data in the necessary text file
direction	string	Takes the required values from the Direction.txt file
membnumberlength	bool	Takes the returned value of the checkmembnumber function
alphamember	bool	Takes the returned value of the checkalphainname function
maxnumber	int	Takes the value of the highest number in the membership number column of the database
newnumb	int	The new membership number generated for a new member

Annotated Program Code:

Home Screen Form Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Membership
{
    public partial class Home_Screen : Form
    {
        public Home_Screen()
        {
            InitializeComponent();
        }

//Algorithm encrypting password
        private string encryptpass()
        {
            string pass = passwordTextBox.Text;
            char[] letter = new char[10];
            int[] x = new int[10];

            for (int i = 0; i <= pass.Length; i++)
            {

                letter[i] = Convert.ToChar(pass.Substring(i, i + 1));
                x[i] = (int)letter[i];
                x[i] = x[i] + 4;
                if ((x[i] > 122) && (x[i] < 127))
                {
                    x[i] = 97 + (x[i] - 123);

                }
            }

            return pass;
        }

//Checks that the membership number entered by the member exists in members table of
the database
      private bool acceptmno()
            {

                int membnum = Convert.ToInt32(membership_NoTextBox.Text);

                memberDataContext context = new memberDataContext();

                var peopledata =
                    from person in context.Members
                    where person.Membership_No == membnum
                    select new { person.Membership_No };
                foreach (var person in context.Members)
```

```
                {
                    person.Membership_No = membnum;
                }
                return true;
            }
        //Checks that the password entered by the member exists in the members table
of the database
        private bool acceptmpass()
            {
                memberDataContext context = new memberDataContext();
                var peopledata =
                    from person in context.Members
                    where person.Password == passwordTextBox.Text
                    select new { person.Password };
                foreach (var person in context.Members)
                {
                    person.Password = passwordTextBox.Text;
                }
                return true;
            }

        //Checks that the login number entered by a staff member exists in the
personnel table of the database
       private bool acceptplogin()
        {
            int loginno = Convert.ToInt32(membership_NoTextBox.Text);

            PersonnelDataContext context = new PersonnelDataContext();
            var peopledata =
                from person in context.Personnels
                where person.Login_No == loginno
                select new { person.Login_No };
            foreach (var person in context.Personnels)
            {
                person.Login_No = loginno;
            }

            return true;

        }

        //Checks that the password entered by a staff member exists in the personnel
table of the database
       private bool acceptppass()
       {
           PersonnelDataContext context = new PersonnelDataContext();
           var peopledata =
               from person in context.Personnels
               where person.Password == passwordTextBox.Text
               select new { person.Password };
           foreach (var person in context.Personnels)
           {
               person.Password = passwordTextBox.Text;
           }

           return true;
       }

        private void Logon_Click(object sender, EventArgs e)
       {
            //Boolean varaibles taking the return values of the functions of the user
entering data
```

```
            bool acptmembnum = acceptmno();
            bool acptmempass = acceptmpass();
            bool acptstafflog = acceptplogin();
            bool acptstaffpass = acceptppass();

            int membnum = Convert.ToInt32(membership_NoTextBox.Text);


            //Checks there is data present in the membership number text box
                if (membership_NoTextBox.Text == " ")
                {
                    MessageBox.Show("Please enter your membership number");
                }
                else
                    //Checks there is data present in the password text box
                    if (passwordTextBox.Text == " ")
                    {
                        MessageBox.Show("Please enter your password");
                    }
                    else
                        //Checks that the number entered is the correct length
                        if ((membership_NoTextBox.Text.Length) > 8)
                        {
                            MessageBox.Show("Please ensure your membership number or
login number has the correct number of digits");
                        }
                        else
                            if ((membership_NoTextBox.Text.Length) < 7)
                            {
                                MessageBox.Show("Please ensure your membership number
or login number has the correct number of digits");
                            }
                            else
                                //If data is in a valid format and the appropriate
boolean variables are true then the relevant form is opened
                                //Error messages are shown depending on whether the
data is recognised from the database
                if (acptmembnum == true)
                {
                    if (acptmempass == true)
                   {
                    this.Hide();
                    Menu menu = new Menu();
                    menu.Closed += (s, args) => this.Close();
                    menu.Show();
                    }

                    else
                    {
                        MessageBox.Show("The password entered is not recognised for
this account");
                    }
                }
                else
                {
                    MessageBox.Show("The membership number entered is not
recognised");
                }

                if(acptstafflog == true)
                {
                    if (acptstaffpass == true)
```

```
                    {
                    this.Hide();
                    Personnel_Control_Form staff = new Personnel_Control_Form();
                    staff.Closed += (s, args) => this.Close();
                    staff.Show();
                    }
                    else
                    {
                        MessageBox.Show("The password entered is not recognised for
this account");
                    }
                }
                else
                {
                    MessageBox.Show("The login number entered is not recognised");
                }

            }

//When a new member clicks the new member button, the form opens
        private void NewMember_Click(object sender, EventArgs e)
        {
            this.Hide();
            New_Member newmemb = new New_Member();
            newmemb.Closed += (s, args) => this.Close();
            newmemb.Show();
        }
```

Menu Form Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Membership
{

public partial class Menu : Form
    {
        public Menu()
        {
            InitializeComponent();
        }

        private void PurchaseTickets_Click(object sender, EventArgs e)
        {
            this.Hide();
            Ticket_Purchase tickets = new Ticket_Purchase();
            tickets.Closed += (s, args) => this.Close();
            tickets.Show();
        }
        //Opens form for user to purchase tickets
        private void PlanAwayRoute_Click(object sender, EventArgs e)
        {
            this.Hide();
            Route route = new Route();
            route.Closed += (s, args) => this.Close();
            route.Show();
        }
        //Opens form for user to plan route to away game
        private void EmailClub_Click(object sender, EventArgs e)
        {
            this.Hide();
            Email_Page mail = new Email_Page();
            mail.Closed += (s, args) => this.Close();
            mail.Show();
        }
        //Opens form for user to send an e-mail
        private void Logout_Click(object sender, EventArgs e)
        {
            this.Hide();
            Home_Screen home = new Home_Screen();
            home.Closed += (s, args) => this.Close();
            home.Show();
        }
        //Sends user back to home screen
    }
```

Ticket Purchase Form Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Xml.Linq;

namespace Membership
{
    public partial class Ticket_Purchase : Form
    {
        public Ticket_Purchase()
        {
            InitializeComponent();
        }

        //Combo boxes on form are populated with quantity options
        private void Ticket_Purchase_Load(object sender, EventArgs e)
        {
            MembersTicket.Items.Add("0");
            MembersTicket.Items.Add("1");
            MembersTicket.Items.Add("2");
            MembersTicket.Items.Add("3");
            MembersTicket.Items.Add("4");
            MembersTicket.Items.Add("5");

            AdultTicket.Items.Add("0");
            AdultTicket.Items.Add("1");
            AdultTicket.Items.Add("2");
            AdultTicket.Items.Add("3");
            AdultTicket.Items.Add("4");
            AdultTicket.Items.Add("5");

            U18Ticket.Items.Add("0");
            U18Ticket.Items.Add("1");
            U18Ticket.Items.Add("2");
            U18Ticket.Items.Add("3");
            U18Ticket.Items.Add("4");
            U18Ticket.Items.Add("5");
        }

        //Checks to see if a quantity has been selected in the members menu
        private bool checkmembersselected()
        {
            bool selected = false;
            if (MembersTicket.SelectedIndex > -1)
            {
                selected = true;
            }
            return selected;
        }

        //Checks to see if a quantity has been selected in the adult menu
        private bool checkadultselected()
        {
            bool selected = false;
```

```
            if (AdultTicket.SelectedIndex > -1)
            {
                selected = true;
            }
            return selected;
        }

        //Checks to see if a quantity has been selected in the under 18 menu
        private bool checku18selected()
        {
            bool selected = false;
            if (U18Ticket.SelectedIndex > -1)
            {
                selected = true;
            }
            return selected;
        }

        //Gets the number of member tickets the member wants to purchase
        public int getmemberquantity()
        {
            int quantity = Convert.ToInt32(MembersTicket.SelectedValue);
            return quantity;
        }

        //Gets the number of adult tickets the member wants to purchase
        public int getadultquantity()
        {
            int quantity = Convert.ToInt32(AdultTicket.SelectedValue);
            return quantity;
        }

        //Gets the number of U18 tickets the member wants to purchase
        public int getu18quantity()
        {
            int quantity = Convert.ToInt32(U18Ticket.SelectedValue);
            return quantity;
        }

        //Calculates the total cost of the purchase for the member
        public int CalcCost()
        {
            int adultquantity = getadultquantity();
            int membersquantity = getmemberquantity();

            int totalcost;
            totalcost = ((adultquantity * 9) + (membersquantity * 5));
            return totalcost;
        }

        //Method writing the PayPal html file
        private void writepayhtml()
        {
            int totalcost = CalcCost();
            string[] htmllines = { "<form action=\"https//www.sandbox.paypal.com/cgi-
bin/webscr\" method=\"post>",
                                   " ",
                                   "<input type=\"hidden\" name=\"cmd\"
value=\"_xclick\" />",
                                   "<input type=\"hidden\" name=\"business\"
value=\"hannah.short1@btinternet.com\" />",
                                   " ",
```

```
                                "<input type=\"hidden\" name=\"item_name\"
value=\"Total Cost\" />",
                                "<input type=\"hidden\" name=\"amount\" value=\"" +
totalcost + "\" /> ",
                                "<input type=\"hidden\" name=\"currency_code\"
value=\"GBP\">",
                                "<input type=\"submit\" value=\"Buy\" />",
                                " ",
                                "</form>"
                            };

System.IO.File.WriteAllLines(@"C:\Users\\Owner\Documents\College\Computing\pay.html ",
htmllines);

        }

        private void Pay_Click(object sender, EventArgs e)
        {
            bool memberselect = checkmembersselected();
            bool adultselect = checkadultselected();
            bool u18select = checku18selected();

            if (memberselect == false)
            {
                MessageBox.Show("Please select a quantity of member tickets");
            }
            else
                if (adultselect == false)
                {
                    MessageBox.Show("Please select a quantity of adult tickets");
                }
                else
                    if (u18select == false)
                    {
                        MessageBox.Show("Please select a quantiy of under 18
tickets");
                    }
                    else
                    {
                        writepayhtml();
                        System.Diagnostics.Process.Start("h:/Computing/Unit
4/pay.html");
                        this.Hide();
                        Ticket_Confirmation tconfirm = new Ticket_Confirmation();
                        tconfirm.FormClosed += (s, args) => this.Close();
                        tconfirm.Show();
                    }
        }

        private void Logout_Click(object sender, EventArgs e)
        {
            this.Hide();
            Home_Screen home = new Home_Screen();
            home.FormClosed += (s, args) => this.Close();
            home.Show();
        }

        private void Back_Click(object sender, EventArgs e)
        {
            this.Hide();
            Menu menu = new Menu();
```

```
            menu.FormClosed += (s, args) => this.Close();
            menu.Show();
        }


    }
}
```

HTML Code used in form:

```
<form          action="https://www.sandbox.paypal.com/cgi-bin/webscr"
method="post">

    <input type="hidden" name="cmd" value="_xclick" />
    <input                type="hidden"                name="business"
value="hannah.short1@btinternet.com" />

    <input type="hidden" name="item_name" value="Total Cost" />
    <input type="hidden" name="amount" value="*purchase value*" />
      <input type="hidden" name="currency_code" value="GBP">

</form>
```

Route Form Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Membership
{
    public partial class Route : Form
    {
        public Route()
        {
            InitializeComponent();
        }

        //Combo box is populated with Ryman Youth league teams when form loads
        private void Route_Load(object sender, EventArgs e)
        {
            Teams.Items.Add("Bognor Regis Football Club");
            Teams.Items.Add("Eastbourne Borough Football Club");
            Teams.Items.Add("Eastbourne Town Football Club");
            Teams.Items.Add("Hastings United Football Club");
            Teams.Items.Add("Horsham Football Club");
            Teams.Items.Add("Lewes Football Club");
            Teams.Items.Add("Three Bridges Football Club");
            Teams.Items.Add("Whitehawk Football Club");
            Teams.Items.Add("Worthing Football Club");
        }

        //Gets the team the user has selected
        public string getselectedteam()
        {
            string club = Convert.ToString(Teams.SelectedValue);
            return club;
        }

        //Gets the corresponding node of the team on the graph
        private int getnode()
        {
            string team = getselectedteam();
            int node;
            switch (team)
            {
                case "Bognor Regis Football Club":
                    node = 1;
                    break;
                case "Eastbourne Borough Football Club":
                    node = 2;
                    break;
                case "Eastbourne Town Football Club":
                    node = 3;
                    break;
                case "Hastings United Football Club":
                    node = 4;
                    break;
                case "Horsham Football Club":
```

```
                node = 5;
                break;
            case "Lewes Football Club":
                node = 6;
                break;
            case "Three Bridges Football Club":
                node = 7;
                break;
            case "Whitehawk Football Club":
                node = 8;
                break;
            case "Worthing Football Club":
                node = 9;
                break;
            default:
                node = 0;
                break;
        }
        return node;
    }

    //Algorithm finding the node on the graph the member has selected
    private bool calcroute(int Node, int EndNode)
    {
        int[,] weights = new int[,]
            {
        {0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,1,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},
        {1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,1,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,1,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0},
        };

        bool[] Found = new bool[30];
        bool[] DestinationReached = new bool[30];
        bool Reached = false;

        if (Node == EndNode)
        {
            Reached = true;
        }
        for (int i = 0; i < 30; i++)
        {
            if (Found[i] == false)
            {
                calcroute(Node, EndNode);
            }
        }
        DestinationReached[Node] = true;
```

```
            return DestinationReached[Node];
        }

        //Method checking if the miles checkbox has been checked
        public bool MilesSelected()
        {
            bool check = false;
            if (Miles.Checked == true)
            {
                check = true;
            }
                return check;
        }

        //Method checking if the kilometres checkbox has been checked
        public bool KMSelected()
        {
            bool check = false;
            if (KM.Checked == true)
            {
                check = true;
            }
                return check;
        }

        private void GetRoute_Click(object sender, EventArgs e)
      {
          bool milesselected = MilesSelected();
          bool kmselected = KMSelected();

            //An error is given if no team has been selected
            if (Teams.SelectedIndex <= -1)
            {
                MessageBox.Show("Please ensure you have selected a team");
            }
            else
                //An error is given if neither the miles or kilometres options have
been selected
                if ((milesselected = false) && (kmselected == false))
                {
                    MessageBox.Show("Please check you have selected if you want your
journey to be displayed in miles or kilometres");
                }
                else
                    //An error is given if both options have been selected
                    if ((milesselected = true) && (kmselected == true))
                    {
                        MessageBox.Show("Please check you have only selected one
distance option");
                    }
                    else
                        //If the options selected are valid, the route plan is loaded
                {
                this.Hide();
                Route_Plan r = new Route_Plan();
                r.Closed += (s, args) => this.Close();
                r.Show();
                }
        }

        private void Logout_Click(object sender, EventArgs e)
```

```
        {
            this.Hide();
            Home_Screen home = new Home_Screen();
            home.Closed += (s, args) => this.Close();
            home.Show();
        }

        private void Back_Click(object sender, EventArgs e)
        {
            this.Hide();
            Menu menu = new Menu();
            menu.Closed += (s, args) => this.Close();
            menu.Show();
        }
    }
}
```

E-mail Form Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Net;
using System.Net.Mail;
using System.Text;
using System.Threading.Tasks;

using System.Windows.Forms;

namespace Membership
{
    public partial class Email_Page : Form
    {
        public Email_Page()
        {
            InitializeComponent();
        }

//Checks that all the text boxes are full
        private bool checkdatapresent()
        {
            bool presentdata = false;
            if (fNameTextBox.Text == " ")
            {
                MessageBox.Show("Please enter your first name");
            }
            else
                if (sNameTextBox.Text == " ")
                {
                    MessageBox.Show("Please enter your surname");
                }
                else
                    if (phonetextBox.Text == " ")
                    {
                        MessageBox.Show("Please enter a phone number");
                    }
                    else
                        if (addresstextBox.Text == " ")
                        {
                            MessageBox.Show("Please enter your e-mail address");
                        }
                        else
                            if (subjecttextBox.Text == " ")
                            {
                                MessageBox.Show("Please enter a subject");
                            }
                            else
                                if (messagetextBox.Text == " ")
                                {
                                    MessageBox.Show("Please enter a message");
                                }
                                else
                                {
                                    presentdata = true;
                                }
```

```
            return presentdata;
        }

        //Checks there are no alpha characters in the phone number
        private bool checkalphanumber()
        {
            bool IsAlpha = false;
            foreach (char c in phonetextBox.Text)
            {
                if (!Char.IsNumber(c))
                {
                    IsAlpha = true;
                    break;
                }
            }

            return IsAlpha;
        }

        //Checks the range of the subject
        private bool checksubjectlength()
        {
            bool length = false;
            if (subjecttextBox.Text.Length <= 40)
            {
                length = true;
            }
            return length;
        }

        //Checks the range of the message
        private bool checkmessagelength()
        {
            bool mlength = false;
            if (messagetextBox.Text.Length <= 400)
            {
                mlength = true;
            }
            return mlength;
        }

        private void Send_Click(object sender, EventArgs e)
        {
            bool datapresent = checkdatapresent();
            bool presentalpha = checkalphanumber();
            bool validsubjectlength = checksubjectlength();
            bool validmessagelength = checkmessagelength();

            //If the checkalphanumber method finds letters in the phone number, an
error message is displayed
            if (presentalpha == true)
            {
                MessageBox.Show("Ensure your phone number contains only numeric
characters");
            }
            else
                //If the subject length is over 40 characters, there is an error
                if (validsubjectlength == false)
                {
                    MessageBox.Show("Ensure your subject is less than 40 characters");
                }
                else
```

```
                    if (validmessagelength == false)
                    {
                        MessageBox.Show("Ensure your message is less than 400
characters");
                    }
                    else
                    //If all data is present, the e-mail gets sent using smtp protocol
                    if (datapresent == true)
                    {
                        try
                        {
                            //E-mail message constructed from data entered
                            MailMessage message = new MailMessage();
                            message.To.Add("hannah.short1@btinternet.com");
                            string mailaddress;
                            mailaddress = addresstextBox.Text;
                            MailAddress address = new MailAddress(mailaddress);
                            message.From = address;
                            string subject = subjecttextBox.Text;
                            message.Subject = subject;
                            string msg = messagetextBox.Text;
                            message.Body = msg;

                            var smtp = new System.Net.Mail.SmtpClient();
                            {
                                smtp.Host = " ";
                                smtp.Port = 587;
                                smtp.EnableSsl = true;
                                smtp.DeliveryMethod =
System.Net.Mail.SmtpDeliveryMethod.Network;
                            }
                            //An e-mail confirmation form comes up when the user has
clicked send
                            this.Hide();
                            Emal_Confirmation confirm = new Emal_Confirmation();
                            confirm.Closed += (s, args) => this.Close();
                            confirm.Show();
                        }
                            //If the message isn;t succesfully sent, there is a
default error message
                        catch { MessageBox.Show("Error Sending E-mail, Please try
again"); }
                    }

        }

        private void membersBindingNavigatorSaveItem_Click(object sender, EventArgs e)
        {
            this.Validate();
            this.membersBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.membersDataSet);

        }

        //Takes user back to home screen
        private void Logout_Click(object sender, EventArgs e)
        {
            this.Hide();
            Home_Screen home = new Home_Screen();
            home.FormClosed += (s, args) => this.Close();
            home.Show();
        }
```

```
        //Takes user back to menu
        private void Back_Click(object sender, EventArgs e)
        {
            this.Hide();
            Menu menu = new Menu();
            menu.FormClosed += (s, args) => this.Close();
            menu.Show();
        }
    }
   }
```

Ticket Confirmation Form Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Membership
{
    public partial class Ticket_Confirmation : Form
    {
        public Ticket_Confirmation()
        {
            InitializeComponent();
        }
            int adultticket;
            int memberticket;
            int u18ticket;

        private Ticket_Purchase tp;

        //Gets the ticket quantities selected on the previous form
        public void getticketquantities()
        {
            adultticket = tp.getadultquantity();
            memberticket = tp.getmemberquantity();
            u18ticket = tp.getu18quantity();

        }

        //The label confirming the tickets bought by the user is written when the form
loads
        private void Ticket_Confirmation_Load(object sender, EventArgs e)
        {
            TConfirm.Text = "Thank you for puchasing your tickets.\r\nYou have
purchased:\r\n" +  adultticket + "adult tickets,\r\n" + memberticket + "members
tickets,\r\n" + u18ticket + "under 18 tickets";
        }
    }
}
```

Route Plan Form Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Membership
{
    public partial class Route_Plan : Form
    {

        public Route_Plan()
        {
            InitializeComponent();

        }

        string club;
        private Route r;

        //A method getting the team name from the previous form
        private void getteam()
        {
            club = r.getselectedteam();
        }

        private void Route_Plan_Load(object sender, EventArgs e)
        {
            bool mileschecked = r.MilesSelected();
            bool kmchecked = r.KMSelected();

            //The club selected on the previous form is written into the first text
box on the form
            ClubName.Text = club;

            //If the miles option was selected on the previous form
            if (mileschecked == true)
            {
                string distance =
System.IO.File.ReadAllText(@"f:/Membership/DistanceMiles.txt");
                Distance.Text = distance;
           }
            else
            if (kmchecked == true)
            {
                string distance =
System.IO.File.ReadAllText(@"f:/Membership/DistanceKM.txt");
                Distance.Text = distance;
            }

            string direction = System.IO.File.ReadAllText(@"f:/Membership/Directions
.txt");
            Direction.Text = direction;
```

```
        }

        private void Logout_Click(object sender, EventArgs e)
        {
            this.Hide();
            Home_Screen home = new Home_Screen();
            home.Closed += (s, args) => this.Close();
            home.Show();
        }

        private void Back_Click(object sender, EventArgs e)
        {
            this.Hide();
            Menu menu = new Menu();
            menu.Closed += (s, args) => this.Close();
            menu.Show();
        }

        private void Direction_TextChanged(object sender, EventArgs e)
        {
            Direction.ScrollBars = ScrollBars.Vertical;
        }

        private void Distance_TextChanged(object sender, EventArgs e)
        {
            Distance.ScrollBars = ScrollBars.Vertical;
        }



    }
}
```

E-mail Confirmation Form Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Membership
{
    public partial class Emal_Confirmation : Form
    {
        public Emal_Confirmation()
        {
            InitializeComponent();
        }

        //The user is able to logout when they have read the confirmation message
        private void Logout_Click(object sender, EventArgs e)
        {
            this.Hide();
            Home_Screen home = new Home_Screen();
            home.FormClosed += (s, args) => this.Close();
            home.Show();
        }

        //When the confiramtion form loads, a label thanking the member is written
        private void Emal_Confirmation_Load(object sender, EventArgs e)
        {
            ConfirmMail.Text = "Thank you for your e-mail.\r\nWe will reply to you
shortly";
        }
    }
}
```

Personnel Control Form Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Membership
{
    public partial class Personnel_Control_Form : Form
    {
        public Personnel_Control_Form()
        {
            InitializeComponent();
        }

        private void membersBindingNavigatorSaveItem_Click(object sender, EventArgs e)
        {
            this.Validate();
            this.membersBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.membersDataSet);

        }

//Algorithm encrypting password
        private string encryptpass()
        {
            string pass = password.Text;
            char[] letter = new char[10];
            int[] x = new int[10];

            for (int i = 0; i <= pass.Length; i++)
            {

                letter[i] = Convert.ToChar(pass.Substring(i, i + 1));
                x[i] = (int)letter[i];
                x[i] = x[i] + 4;
                if ((x[i] > 122) && (x[i] < 127))
                {
                    x[i] = 97 + (x[i] - 123);

                }
            }

            return pass;
        }

//Method checking there is data in all the text boxes and returning false if there
isn't

private bool checkdatapresent()
        {
            bool presentdata = false;
            if (MembNumber.Text == " ")
            {
                MessageBox.Show("Please enter membership number of member");
            }
```

```
            else
                if (password.Text == " ")
                {
                    MessageBox.Show("Please enter a password for member");
                }
                else
            if (fName.Text == " ")
            {
                MessageBox.Show("Please enter your first name");
            }
            else
                if (sName.Text == " ")
                {
                    MessageBox.Show("Please enter your surname");
                }
                else
                    if (address.Text == " ")
                    {
                        MessageBox.Show("Please enter your address");
                    }
                    else
                        if (postcode.Text == " ")
                        {
                            MessageBox.Show("Please enter your postcode");
                        }
                        else
                        {
                            presentdata = true;
                        }

            return presentdata;
        }

//Checks the membership number is the right length
        private bool checkmembnumber()
        {
            bool length = false;

            if (MembNumber.Text.Length == 8)
            {
                length = true;
            }
            return length;
        }

//Checks there are no alpha characters in the membership number entered
        private bool checkalphainname()
        {
            bool IsAlpha = true;
            foreach (char c in MembNumber.Text)
            {
                if (!Char.IsNumber(c))
                {
                    IsAlpha = false;
                    break;
                }
            }
            return IsAlpha;
        }

//Checks that the first name and surname that have been entered don't contain any
numbers
```

```
        private bool checknumericinname()
        {

            bool IsNumeric = true;

            foreach (char c in fName.Text)
            {
                if (!Char.IsNumber(c))
                {
                    IsNumeric = false;
                    break;
                }
            }
            foreach (char d in sName.Text)
            {
                if (!Char.IsNumber(d))
                {
                    IsNumeric = false;
                    break;
                }
            }

            return IsNumeric;
        }

//Checks the postcode is the correct length
        private bool validatepclength()
        {
            bool length = false;
            if (postcode.Text.Length == 7)
            {
                length = true;
            }
            return length;
        }

//Boolean variables taking values of validation methods
        private void AddMember_Click(object sender, EventArgs e)
        {
            //Boolean variables taking values of validation methods
            bool datapresent = checkdatapresent();
            bool membnumberlength = checkmembnumber();
            bool presentnumeric = checknumericinname();
            bool validpc = validatepclength();

            //If the membership number length is not right, there is an error message
            if (membnumberlength == false)
            {
                MessageBox.Show("Ensure the membership number entered is 8
characters")
            }
            else
            //If there are numbers in the names, there is an error
            if (presentnumeric == true)
            {
                MessageBox.Show("Ensure the name entered contains no numeric
characters");
            }
            else
                //If the postcode is the wrong length, there is an error
                if (validpc == false)
                {
```

```
                    MessageBox.Show("Ensure the postcode is the correct number of
characters");
                }
                else
                    //If data entered is valid, the member is added to the members
table
                {
                    memberDataContext context = new memberDataContext();

                    Member m = new Member
                    {
                        FName = fName.Text,
                        SName = sName.Text,
                        Address = address.Text,
                        Postcode = postcode.Text
                    };

                    context.Members.InsertOnSubmit(m);

                    try
                    {
                        context.SubmitChanges();
                    }
                        //There is an exception if the data can't be added
                    catch (Exception e1)
                    {
                        Console.WriteLine(e1);
                        context.SubmitChanges();
                    }
                }
        }


private void DeleteMember_Click(object sender, EventArgs e)
        {
            bool membnumberlength = checkmembnumber();
            bool alphamember = checkalphainname();

            int membnum = Convert.ToInt32(MembNumber.Text);

            //Checks the membership number entered is the right length
            if (membnumberlength == false)
            {
                MessageBox.Show("Ensure the membership number entered is 8
characters");
            }
            else
                //Checks the membership number has no letters in it
                if (alphamember == true)
                {
                    MessageBox.Show("Ensure the membership number entered onlyt
contains numeric characters");
                }
                else
                    //Finds membership number in database and deletes member
                {
                    memberDataContext context = new memberDataContext();

                    var peopledata =
                        from person in context.Members
                        where person.Membership_No == membnum
                        select person;
```

```
                    foreach (var member in peopledata)
                    {
                        context.Members.DeleteOnSubmit(member);
                    }

                    try
                    {
                        context.SubmitChanges();
                    }
                        //Exception if member is not deleted from the table
                    catch (Exception e1)
                    {
                        Console.WriteLine(e1);
                    }

                }
        }

/The details of a member can be overwritten when the member needs to change them
        private void EditMember_Click(object sender, EventArgs e)
        {
            int membnum = Convert.ToInt32(MembNumber.Text);

            memberDataContext context = new memberDataContext();

            var editpeople =
                from person in context.Members
                where person.Membership_No == membnum
                select person;
                foreach (Member person in editpeople)
                {
                    person.Membership_No = membnum;
                    person.Password = password.Text;
                    person.FName = fName.Text;
                    person.SName = sName.Text;
                    person.Address = address.Text;
                    person.Postcode = postcode.Text;
                }

                try
                {
                    context.SubmitChanges();
                }
                catch (Exception e1)
                {
                    Console.WriteLine(e1);
                }

        }

//A member can be searched for and their details are displayed in the text boxes on
the form
        private void SearchMember_Click(object sender, EventArgs e)
        {
            int membnum = Convert.ToInt32(MembNumber.Text);

            memberDataContext context = new memberDataContext();

            var peopledata =
                from person in context.Members
                where person.Membership_No == membnum
```

```
                select person;
            foreach (var person in peopledata)
            {
                person.Membership_No = membnum;
                person.Password = password.Text;
                person.FName = fName.Text;
                person.SName = sName.Text;
                person.Address = address.Text;
                person.Postcode = postcode.Text;
            }

        }

        private void Logout_Click(object sender, EventArgs e)
        {
            this.Hide();
            Home_Screen home = new Home_Screen();
            home.FormClosed += (s, args) => this.Close();
            home.Show();
        }

    }
}
```

New Members Form Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Membership
{
    public partial class New_Member : Form
    {
        public New_Member()
        {
            InitializeComponent();
        }

        //Checks that data has been entered into every box
        private bool checkdatapresent()
        {
            bool presentdata = false;
            if (fNameTextBox.Text == " ")
            {
                MessageBox.Show("Please enter your first name");
            }
            else
                if (sNameTextBox.Text == " ")
                {
                    MessageBox.Show("Please enter your surname");
                }
                else
                    if (addressTextBox.Text == " ")
                    {
                        MessageBox.Show("Please enter your address");
                    }
                    else
                        if (postcodeTextBox.Text == " ")
                        {
                            MessageBox.Show("Please enter your postcode");
                        }
                        else
                        {
                            presentdata = true;
                        }

            return presentdata;
        }

        //Checks that the names entered in the first name and surname boxes contain no
numeric characters
        private bool checknumericinname()
        {

            bool IsNumeric = true;
            foreach (char c in fNameTextBox.Text)
            {
                if (!Char.IsNumber(c))
                {
```

```
                    IsNumeric = false;
                    break;
                }
            }
            foreach (char d in sNameTextBox.Text)
            {
                if (!Char.IsNumber(d))
                {
                    IsNumeric = false;
                    break;
                }
            }

            return IsNumeric;
        }

        //Checks the postcode entered by the user has the correct number of characters
        private bool validatepclength()
        {
            bool length = false;
            if (postcodeTextBox.Text.Length == 7)
            {
                length = true;
            }
            return length;
        }

//New membership number generated for new member
        public int generatemembnumber()
        {
            int maxnumber = 0;
            int newnumb;

            memberDataContext context = new memberDataContext();

            var peopledata =
                from person in context.Members
                where person.Membership_No == maxnumber
                select new { person.Membership_No };

            foreach (var person in context.Members)
            {
                if (person.Membership_No > maxnumber)
                {
                    person.Membership_No = maxnumber;
                }
            }
            newnumb = maxnumber + 1;
            return newnumb;
        }

        private void Submit_Click(object sender, EventArgs e)
        {
            bool datapresent = checkdatapresent();
            bool presentnumeric = checknumericinname();
            bool validpc = validatepclength();

            //If the method checking for numeric characters returns true, there is an
error
            if (presentnumeric == true)
            {
                MessageBox.Show("Ensure your name contains no numeric characters");
```

```
            }
            else
            if (validpc == false)
            {
                MessageBox.Show("Ensure your postcode is the correct number of
characters");
            }
            else
            {
                        //When data is valid, details of user are added to data set
                            memberDataContext context = new memberDataContext();

                            Member m = new Member
                            {
                                 Membership_No = generatemembnumber(),
                                FName = fNameTextBox.Text,
                                SName = sNameTextBox.Text,
                                Address = addressTextBox.Text,
                                Postcode = postcodeTextBox.Text
                            };

                            context.Members.InsertOnSubmit(m);

                            try
                            {
                                context.SubmitChanges();
                            }
                            catch (Exception e1)
                            {
                                Console.WriteLine(e1);
                                context.SubmitChanges();
                            }
            //Once the user details have been added, the new member receives a
confirmation
                            this.Hide();
                            New_Member_Confirmation confirm = new
New_Member_Confirmation();
                            confirm.Closed += (s, args) => this.Close();
                            confirm.Show();

        }

        private void membersBindingNavigatorSaveItem_Click(object sender, EventArgs e)
        {
            this.Validate();
            this.membersBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.membersDataSet);

        }

//Takes user back to home screen
        private void Back_Click(object sender, EventArgs e)
        {
            this.Hide();
            Home_Screen home = new Home_Screen();
            home.Closed += (s, args) => this.Close();
            home.Show();
        }
```

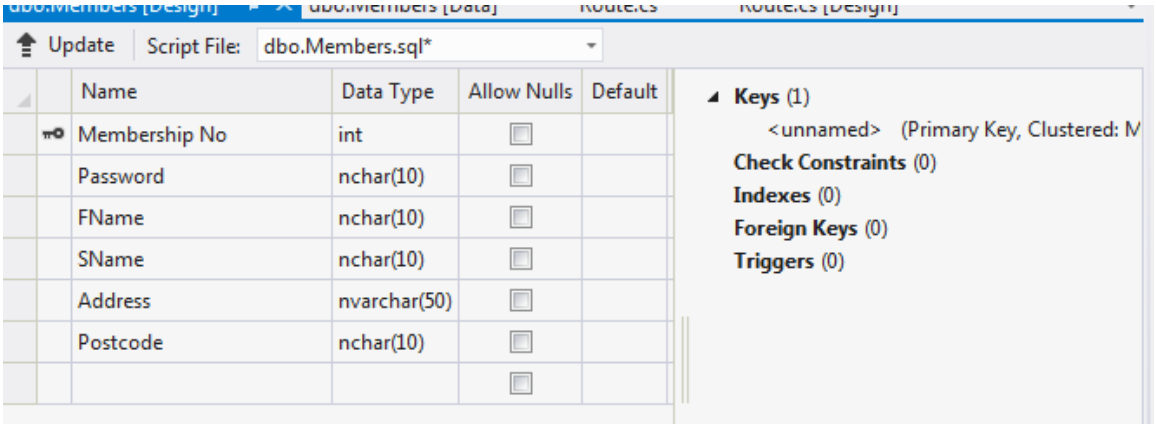
New Members Confirmation Form Code:

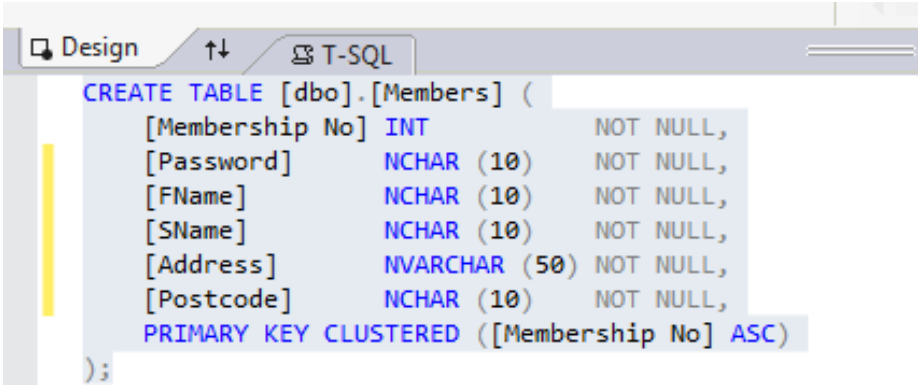
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Membership
{
    public partial class New_Member_Confirmation : Form
    {
        public New_Member_Confirmation()
        {
            InitializeComponent();
        }

        int membnum;
        private New_Member nm;

        //Gets the value from the method on the previous form
        private void getnewnumber()
        {
            membnum = nm.generatemembnumber();
        }

        //Label written when form loadsand generated membership number is written to
text box
        private void New_Member_Confirmation_Load(object sender, EventArgs e)
        {
            ConfirmMessage.Text = "Your details have been added to our\r\nrecords. In
the box below, a unique eight\r\ndigit nmembership number has been\r\ngenerated. You
will need to enter a password\r\nin the password box for your account and\r\nthen
click the submit button. You will then\r\nbe able to login to your account.";
            MembershipNumber.Text = Convert.ToString(membnum);

        }

        //Password encrypted before added to database
        private string encryptpass()
        {
            string pass = PasswordChoice.Text;
            char [] letter = new char[10];
            int[] x = new int[10];

            for (int i = 0; i <= pass.Length; i++)
            {

                letter[i] = Convert.ToChar(pass.Substring(i, i + 1));
                x[i] = (int)letter[i];
                x[i] = x[i] + 4;
                if ((x[i] > 122) && (x[i] < 127))
                {
                    x[i] = 97 + (x[i] - 123);

                }
            }
            return pass;
```

```
        }

        //Password added to database when submit password button is clicked by user
        private void SubmitPass_Click(object sender, EventArgs e)
        {
            string pass = encryptpass();

                memberDataContext context = new memberDataContext();

                Member m = new Member
                {
                    Password = pass
                };


        }
    }
}
```

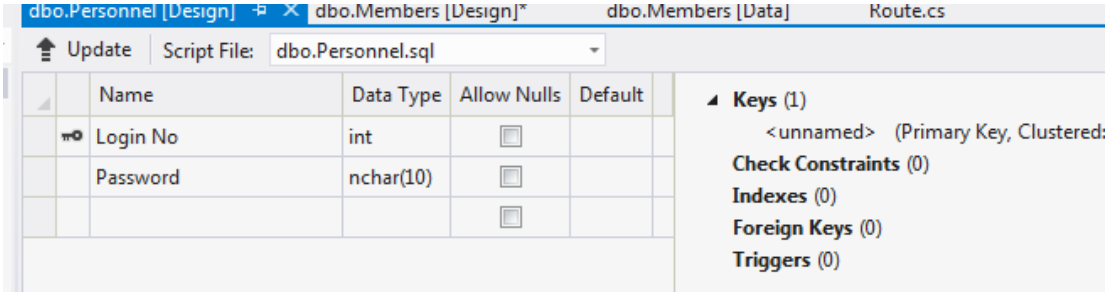
Database Tables:

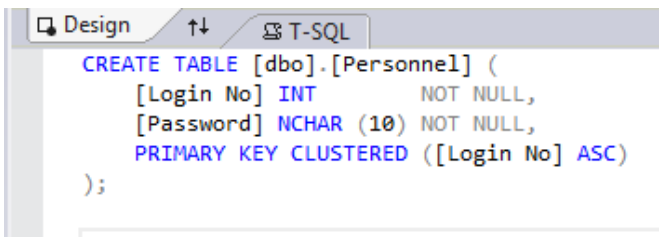
Members Table:



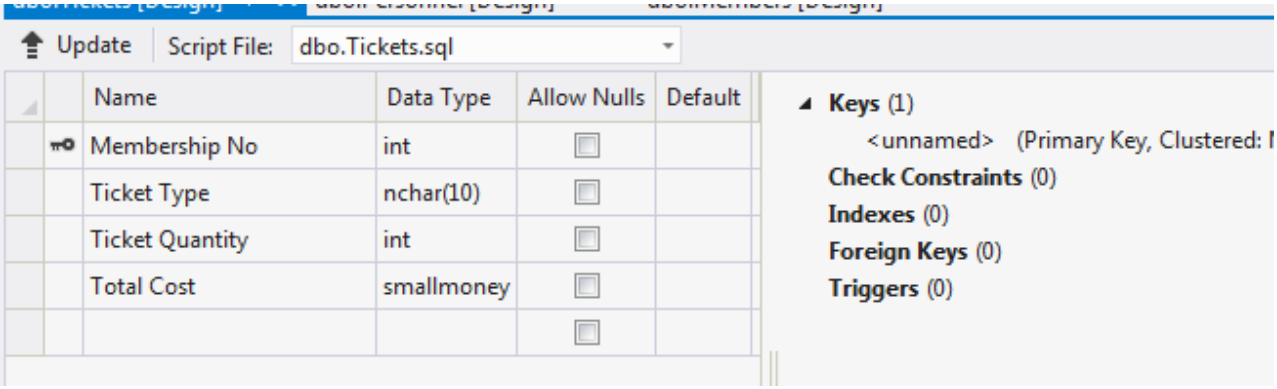


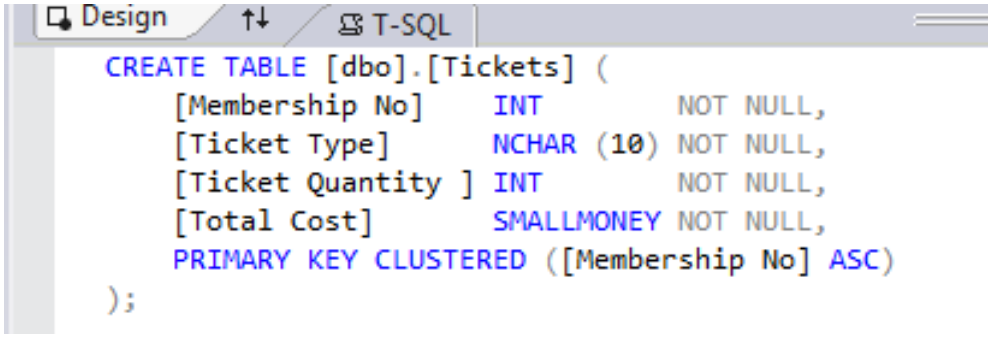
Personnel Table:





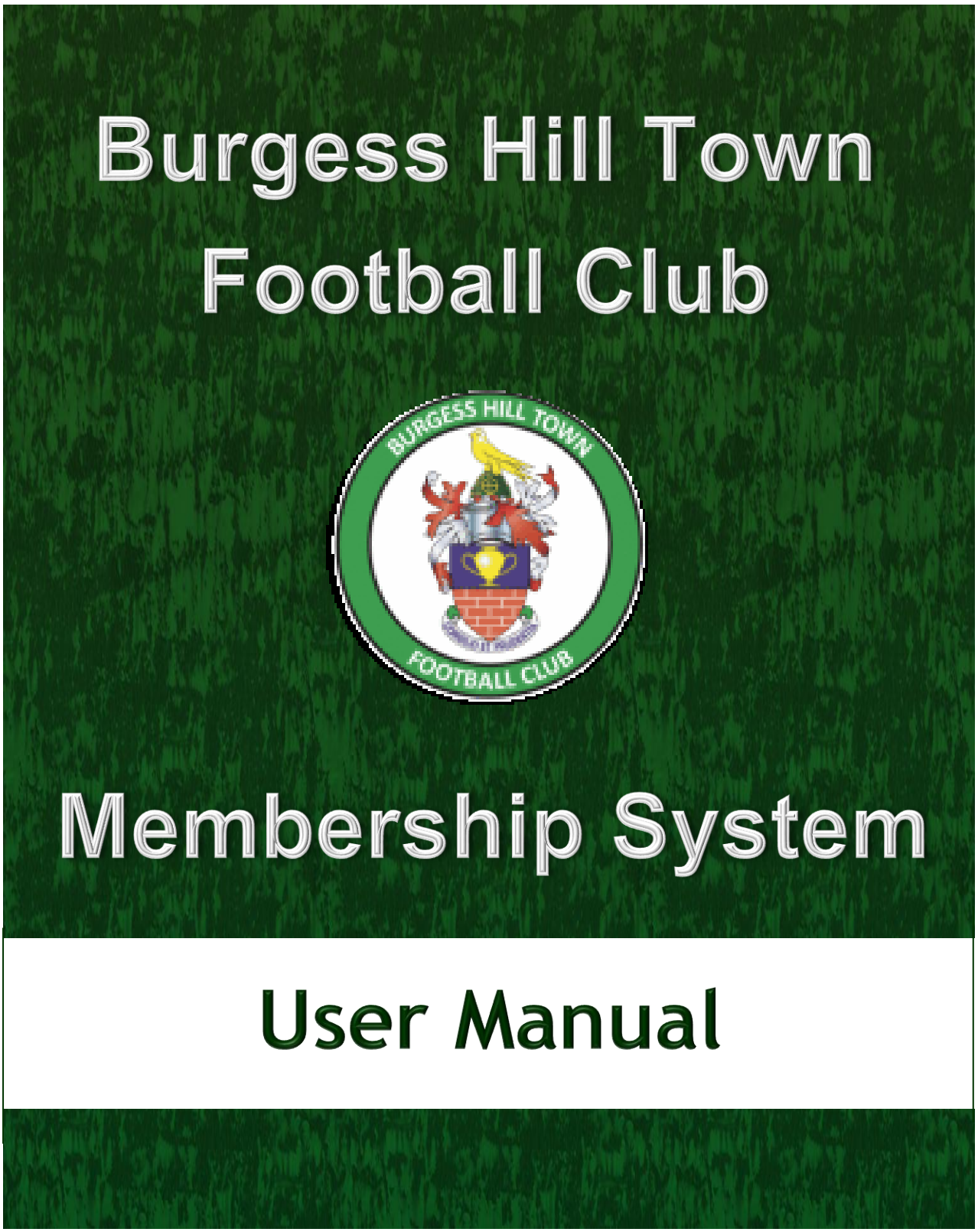
Tickets Table:





User Manual:

Cover Page:



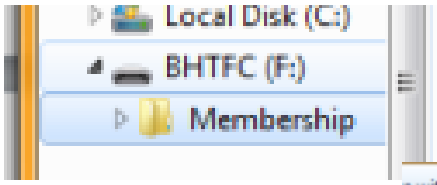
Contents Page:

Contents

Introduction and Installation Instructions:

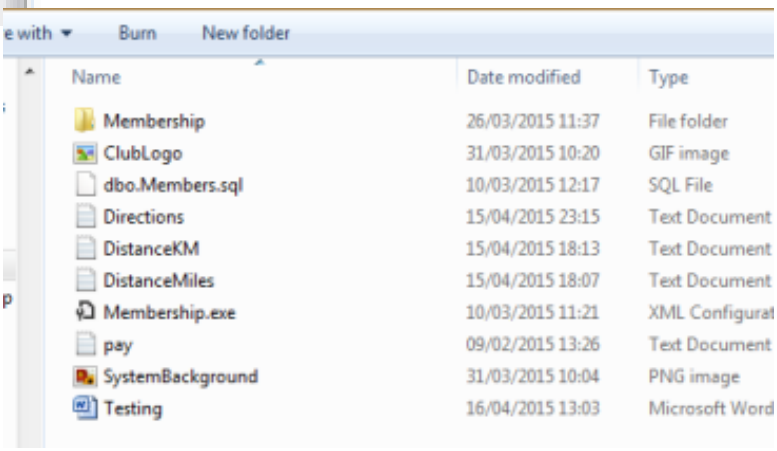
Welcome to the Burgess Hill Football Club Membership System User Manual. This is a guide showing you how to use the different features of the new membership system.

Firstly, you will need to access all the system files and install the system on the club website. On the club USB stick, the folder named 'Membership' will contain all the components which are essential to the running of the system.



The executable version of the system is called 'Membership.exe' and uses the sql database called 'Members.sql'

The text files that the system reads as well as the images used by the system are also stored in this folder.



The folder named 'Membership' within this folder contains all the form data.

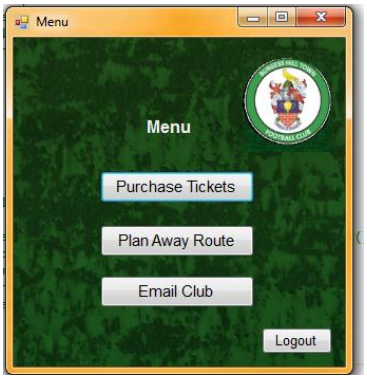
The source code of the club website will need to have an anchor which links users to the executable file contained in the folder.

Use of the System:

Member Login:

When the membership system is first accessed, the home screen is the first thing that the member will see. They will have to enter their club membership number and password which the system will then validate and either output an error message (shown on page 14) or logs them in and direct them to a menu page (see below)

In the first box, the member enters their 8 digit membership number and the second box is for their password. Once they have filled the boxes they can click the logon button. If the data entered is of a valid format, the database of member's details will be checked to ensure that member exists.



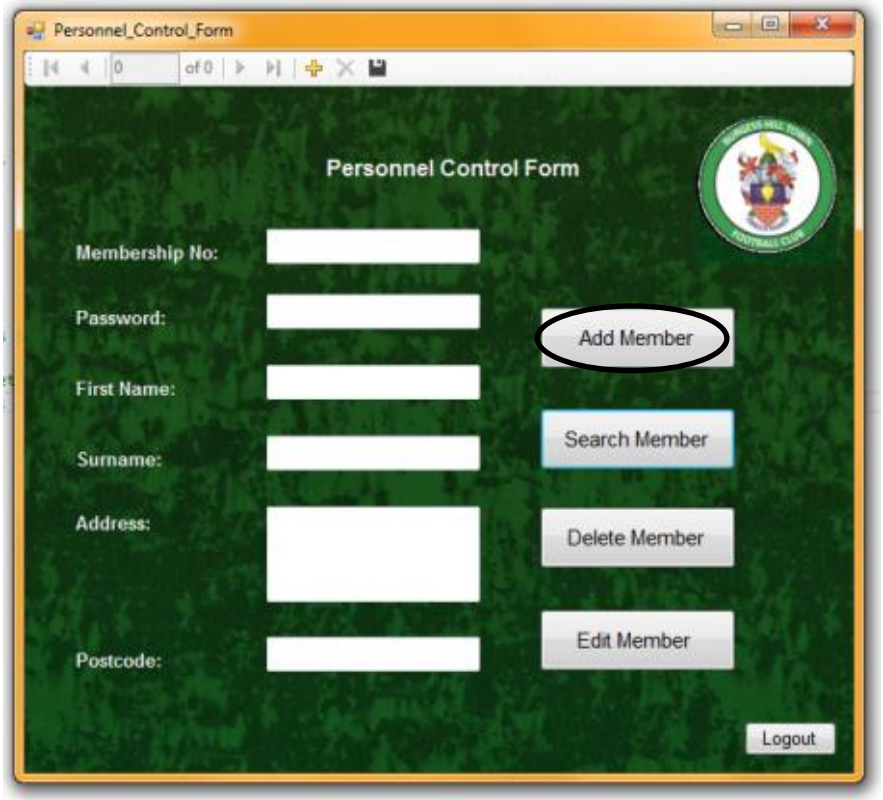
Personnel Login:

A member of staff is able to log on through the home screen as well. If they have a personnel login number and password which is present in the personnel table of the database, they will be able to access a screen where they can manipulate the data of members when they click the logon button.

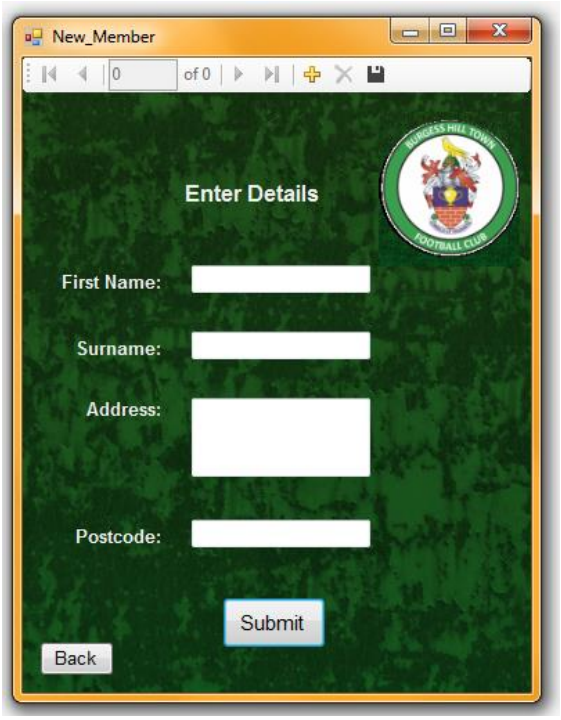
Equally, there will be error messages shown (see page 14) if the number and password entered by club personnel is not valid or doesn't exist.

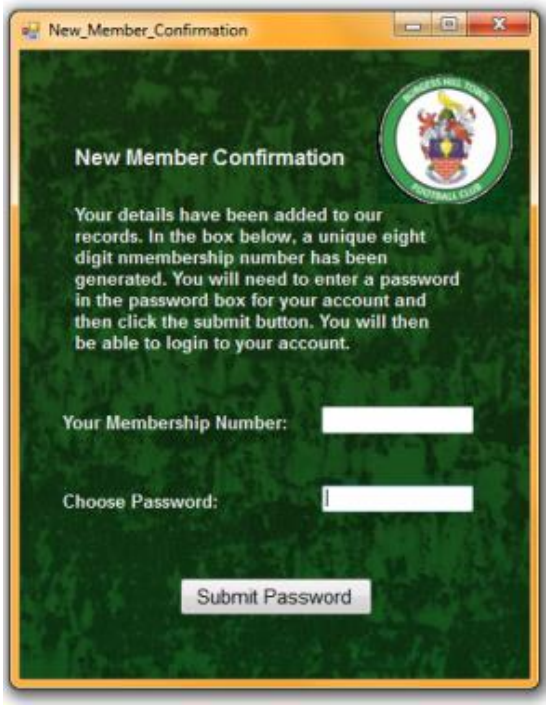
(Right) The personnel control form can be accessed when a staff member logs on and they are able to control the data of the members using this form.

Adding New Members:



A member can be added to the database in one of two ways. Firstly, they are able to be added by a staff member from the personnel control form which is accessed after a member of staff logs into the system. When the text boxes have been completed and the 'Add Member' button has been clicked, a new record will be added to the table of member details. However, the data will be validated to ensure it is of the correct format beforehand and relevant error messages will be shown (see page 14) if it isn't. For instance, if the membership number entered is more than 8 digits long, a message telling the staff member to re-enter the number will be displayed. Screenshots of a member being added can be found on page.

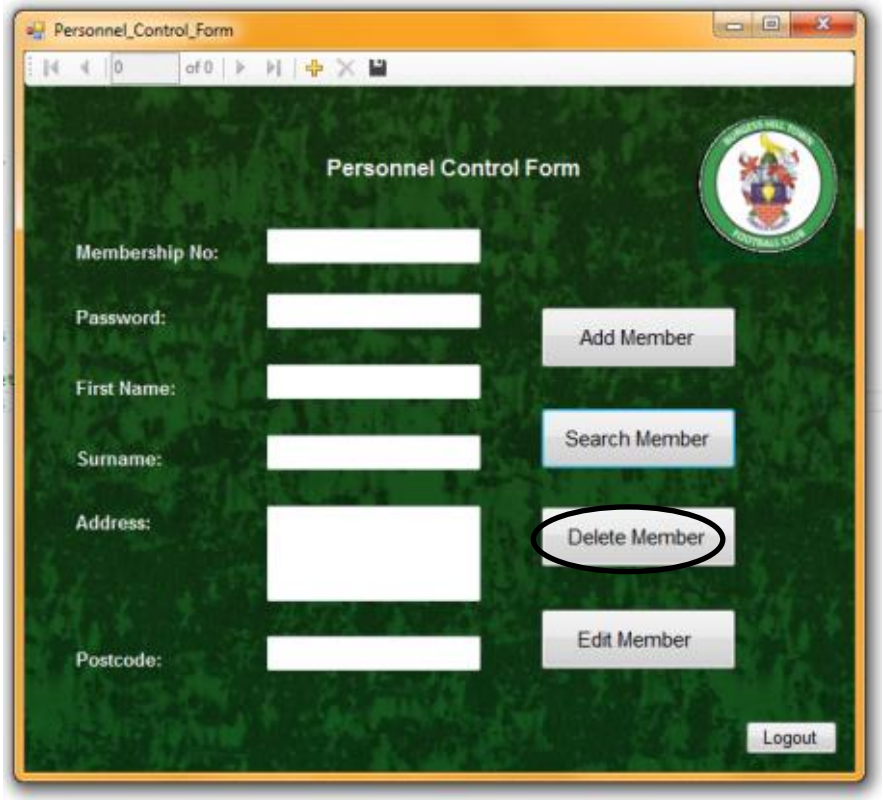




Another way a member can be added is if they do it themselves. On the home screen, there is a button which supporters who don't possess membership can click and a form will open where they are able to enter their first name, surname, address and postcode which will be added to the members table if valid. After, a confirmation form will open where an algorithm will have generated them their unique membership number and written it to the text box labelled with 'Your Membership Number'. In the box below this, they can choose the password they want for their membership account and then click the submit button so the number and password can be added to the table also.

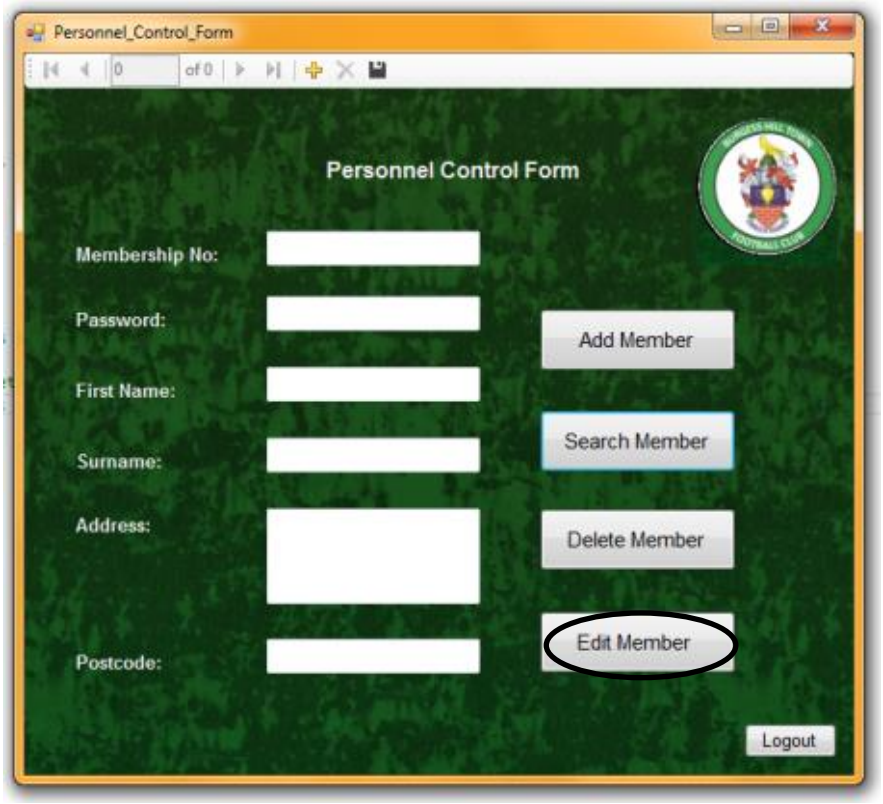
-7-

Deleting Members:



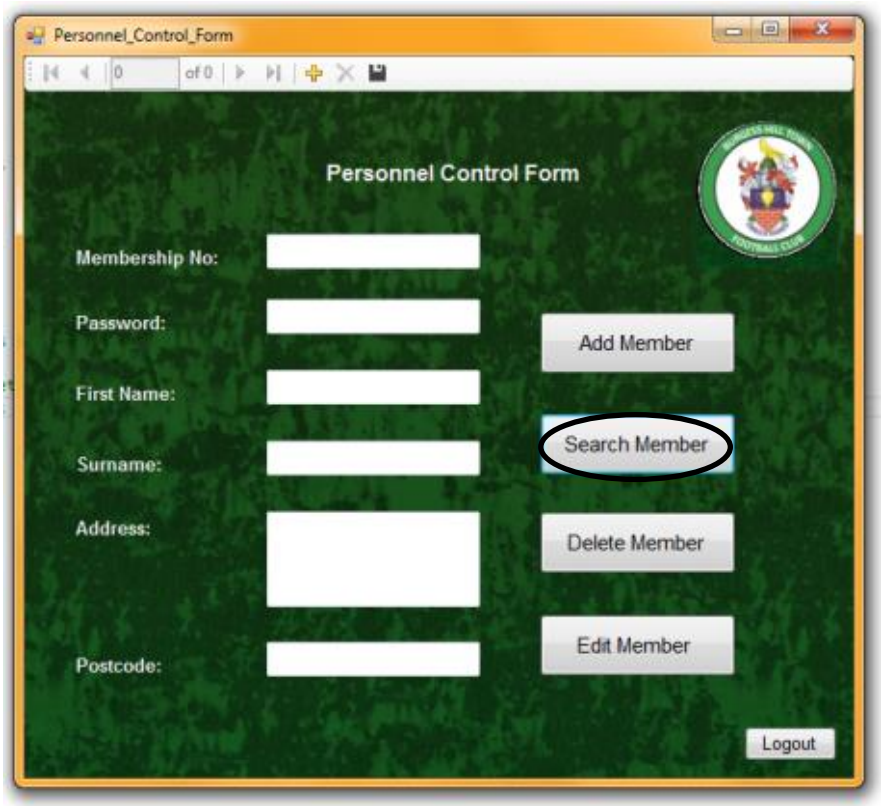
If a member needs to be deleted from the database, on the staff form the 'Delete Member' button can be clicked after the membership number of the member who is going to be removed has been entered into the membership number text box at the top of the form. When the button has been clicked, all their details will be removed from the database.

Editing a Member's Details:



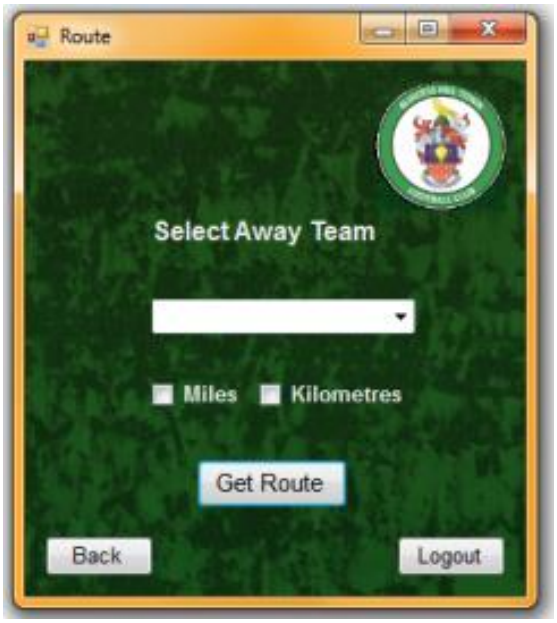
If a member needs some of their details to be altered, all boxes on the personnel control form will need to be filled with the members current details and then when the 'Edit Member' button is clicked, all the data contained in the fields will overwrite the current fields in the database.

Searching Members:



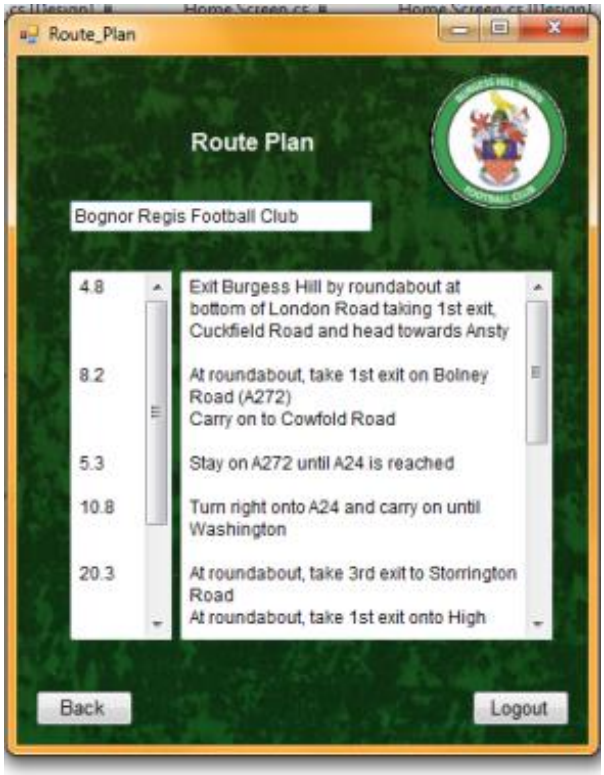
If a staff member needs to look up the details of a member, they can be searched for on the personnel control form. Only one of the boxes needs to be filled before the 'Search Member' button is pressed and the remaining details are displayed. However, it is more useful to enter data into more than one field in the case of members having the same details, e.g. same surname to avoid confusion.

Route Planner Functionality:



From the menu, a member can access the route form. On this form the user will have to select the name of the team to whom they are travelling as well as whether they want their journey to be displayed in either miles or kilometres. When they click the 'Get Route' button, provided they have used the dropdown menu and selected one of the check boxes, they will receive the plan of their route on the next form.

This is the route plan that a member will receive when they have selected the football club they are travelling to and the distance they want their route measured in. In the topmost textbox, the name of the club they have chosen is displayed, in the left hand textbox; there are the distances for each section of the journey and in the final large textbox, the qualitative directions are given and are divided according to each section of the journey. The scrollbars on both of the bottom textboxes can be used to access the entirety of the route due to the size of the form.

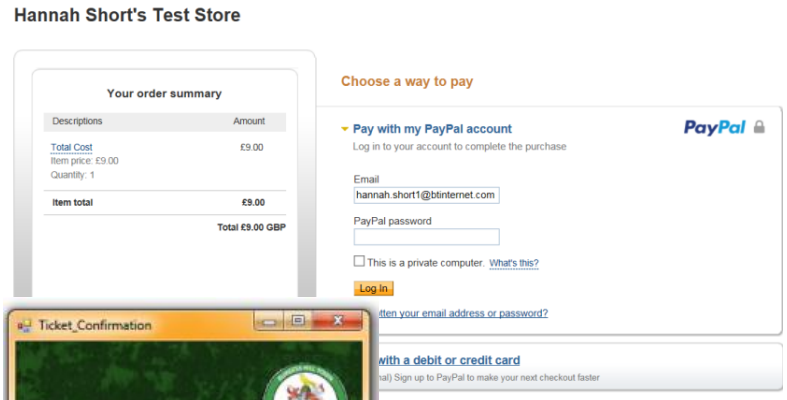


When a user has read the route, they either have the option to go back and select another club or log out of the system taking them back to the home screen

Ticket Purchasing System:

After clicking the 'Purchase Tickets' button on the menu form, the member can access the form on which they can select their quantities of tickets. If nothing has been selected in at least one of the dropdown menus, there will be an error message (see page 14)

When they click the 'Pay' button, they will be directed to an external PayPal page to pay for their tickets. At present, the page is a sandbox test store so you will be required to set up a live PayPal account for the club



A ticket confirmation will also be given to the user with the tickets that they have purchased listed on the form. In the future, the print button on the form will be developed in the future so that users are able to print the form and uses this as confirmation when entering the ground.

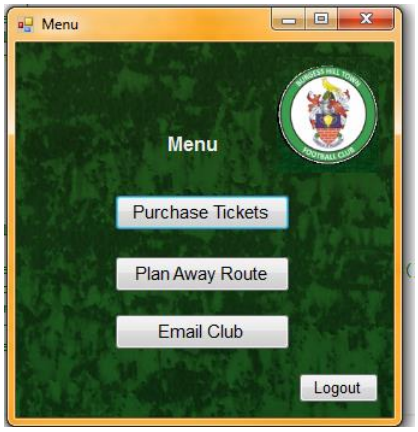
E-mail Guide:

From clicking the 'Email club' button on the menu, users are given access to a form where they are able to enter their name, telephone number, e-mail address into the boxes and a message to send the club in the larger box with an email subject. If any details entered are not of a valid format, there will be an error message depending on the invalidity of the data (see page 14). Note email subjects can only be a maximum of 40 characters while e-mail messages can only be a maximum of 400 characters. Otherwise, if the data entered is all valid, the system will use Simple Mail Transfer Protocol (SMTP) to send the message to the club email account. If the message is routed successfully, there will be an e-mail confirmation thanking the member for their message and informing them the club will reply to them shortly (see below)

System Screenshots:

On the following pages, there are screenshots of the system in use showing how the main components in the system function.

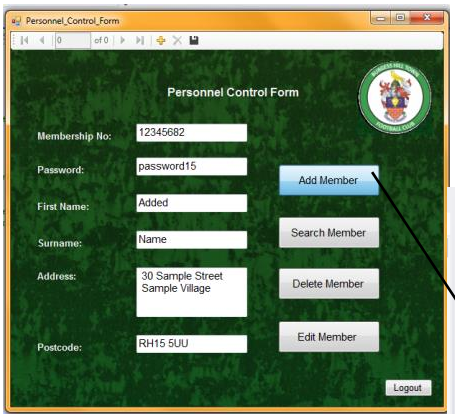
Menu Page:

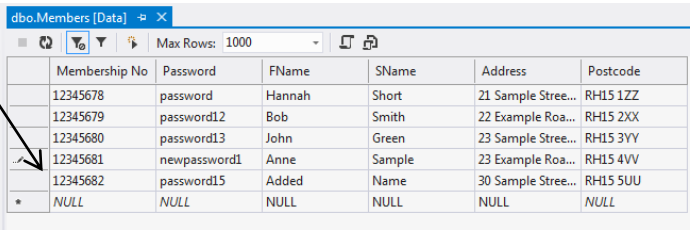


This is the main page that the user is on after they have logged on the home screen; there are three buttons which each link to different sections of the system.

Personnel Control Form:

This is the form where members can be edited. The image shows a member being added to the database once all their details have been entered.





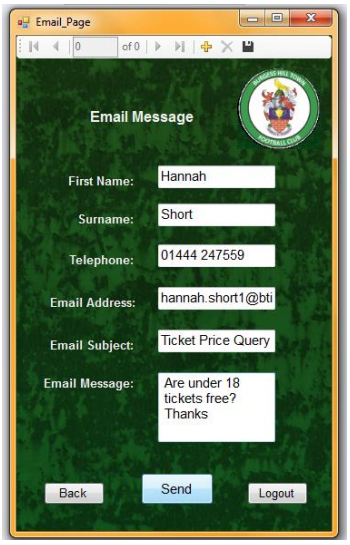
Membership No	Password	FName	SName	Address	Postcode
12345678	password	Hannah	Short	21 Sample Stree...	RH15 1ZZ
12345679	password12	Bob	Smith	22 Example Roa...	RH15 2XX
12345680	password13	John	Green	23 Sample Stree...	RH15 3YY
12345681	newpassword1	Anne	Sample	23 Example Roa...	RH15 4VV
12345682	password15	Added	Name	30 Sample Stree...	RH15 5UU
NULL	*NULL*	NULL	NULL	NULL	*NULL*

Ticket Purchase Form:

The shot of the ticket purchase form shows how a member can select their ticket quantities before clicking the pay button

Route Form:

The shot of the route form shows how a user can select a club of their choice from the menu and then click the get route button to get the route plan



E-mail Form:

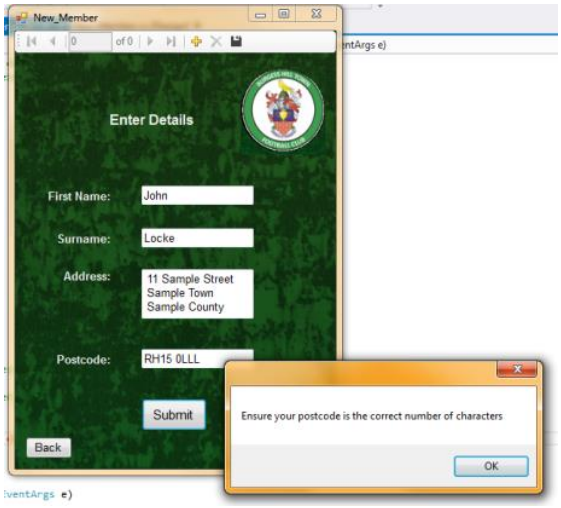
The shot of the e-mail form shows how data can be entered into the text boxes by the user and then the send button can be clicked

Error Message Samples:

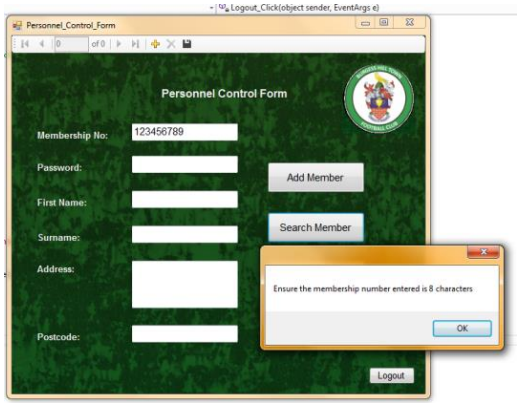
Over the next few pages, there are examples of the type of errors that can occur in the system which have been referenced in the 'Use of the system' section. These show the messages which will be displayed to the user when they enter invalid data, data that is not the correct format or data that doesn't currently exist in the database tables.

When a membership number is entered which doesn't exist in the members table, there is an error informing them

When a staff login number is entered, the personnel table is checked for the presence of the number and if it doesn't exist, there is an error message

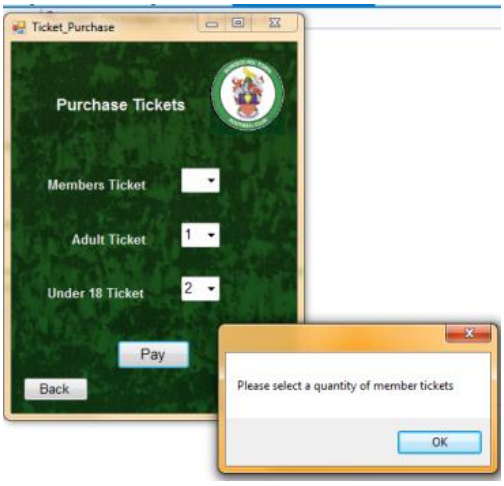


All the details on the new member form are validated by various algorithms. For example, if the length of the postcode entered isn't correct, there is an error message

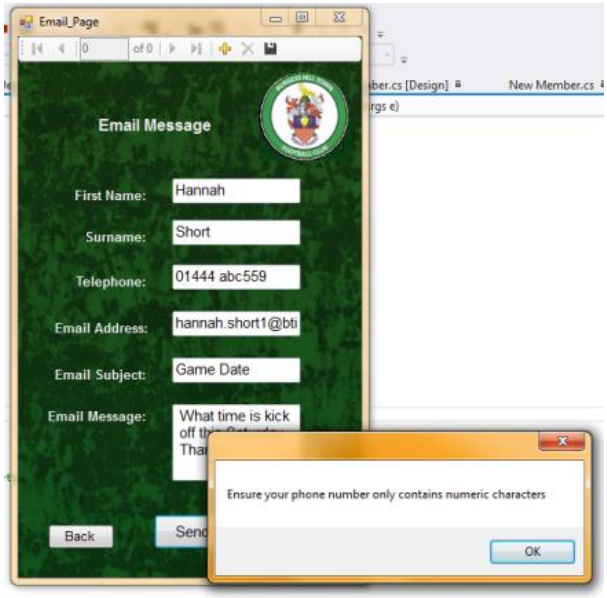


Before a function is executed on the personnel control form, the data entered by the staff member is validated

If boxes are left blank on certain forms, there will be an error asking the user to fill the box or in the case of a dropdown menu, select a data item. For example, on the route form if no club has been selected from the menu, there is an error message displayed to the user



Likewise, when at least one of the menus on the ticket purchase form have been left with no items selected, there is an error



All data entered in the email form is also checked by the system. For instance, if the telephone number entered by the user contains data that is not numeric, there is an error message

Error Recovery Procedures:

1. On the most basic level, when any error messages occur, the user must simply click the 'OK' button on the error message box and re-enter any data that was deemed as invalid. The majority of errors likely to occur on the system will result in this situation
2. If the user finds they are unable to log on to the system although their membership number and password are definitely correct, the club must be contacted to ensure the data they are entering is recorded as theirs and if so necessary measures are taken to ensure they are able to log on
3. If the system is failing to read in the files it uses such as the files used by the route planner, the location of the text files must be checked against the path name used in the system code to ensure it is recognised and altered as necessary
4. If the database functions stop executing as required although data being inputted is valid, the database file must be checked to see whether it is changing and if no plausible issue is found, the system code must be stepped through
5. If no ticket confirmation is given to the user after they have purchased their tickets, the club must be contacted immediately, likewise if a new member receives a confirmation but it fails to contain a membership number

Appraisal:

Comparing Project against Project Objectives:

Original Objectives:

General Objectives:

1. The system must be laid out clearly
2. The system must be easy to navigate around
3. The system must use a similar design scheme as the website
4. Members must have a unique membership number and password
5. The system must cater for new members who want to join the club
6. The system must be made simple for the club personnel to use
7. The system must allow staff members to log in on an account
8. The system must allow staff members to manipulate data
9. The system must allow members to purchase club tickets with credit/debit
10. Members must be able to contact the club via e-mail
11. Members must be provided with a basic route to away grounds when selected

Specific Objectives:

12. The system must provide boxes for people to enter their membership number and password to log on
13. The home screen must have a button for new members to click and enter their details
14. The member must be provided with an error message if they are unsuccessful logging on to the system with a reason why their number and password hasn't worked
15. An error message must be returned to the supporter if their details are invalid
16. The supporters details must be added to the data store if they are all valid
17. Club personnel must be able to securely log on to the system and easily query club member's data when needed; this includes adding, deleting, updating and searching members details
18. Members must be able to select the types and numbers of tickets they want to purchase
19. There must be a secure method of payment for the member to use to buy tickets
20. Members must be provided with a ticket confirmation after a successful purchase
21. There must be an e-mail box on the system for members to write messages; these messages must then be able to be sent to an account created for the club
22. An e-mail confirmation must be provided when the e-mail has been sent successfully
23. Members must be able to select the name of the away team they are going to be travelling to
24. A route must be displayed with basic directions to the chosen away ground

In the following table, I have assessed to what extent I have met each of the original objectives created in the analysis stage of the project:

Key:

✓ = Objective fully met

~ = Objective partially met

X = Objective not met

Objective Number	Objective met?	Comments
1	✓	All the forms and their content have been clearly labelled. For example, the buttons have been given relevant commands and text boxes have been given corresponding labels. When it is more unclear what items are for, further labels have been given to give explanations to the user
2	✓	There is a back button and/or logout button on forms when necessary making it clear to the user how to get back to the home screen of the system or return to the previous form. Also, the buttons to progress with the task the user is performing are labelled clearly so they know what they will be going onto
3	✓	The system uses the same green and white colour scheme as the website does with a green background and white font; Arial has been used, the same as the website. The system uses the official club logo on every form to ensure it looks like official club property
4	✓	Each member stored is identified by their unique membership number and have the opportunity to choose their own password when they are added to the database
5	✓	On the home screen, there is an option new members can access where they can enter their basic details and these will be added to the database
6	✓	When club personnel are using the system, there is a unique control form they can access in order to manipulate the data of the members
7	✓	Each staff member that uses the system will have a login number and a password which will be validated when they log in on the home screen
8	✓	When a staff member has logged on, they can access a form whereby they can either add, delete, edit or search the data of a particular member by clicking the appropriate button

Objective Number	Objective met?	Comments
9	-	From the quantities selected on the ticket purchase form, the total amount the user s required to pay is listed on the PayPal page and the member can log in provided they have a PayPal account. However, there is no current way the ticket confirmation form can open the moment the member buys their ticket on their card
10	x	The e-mail function on the system cannot currently send e-mails to the test account but an error message is not displayed saying the message failed to send
11	✓	Depending on the team the user has selected from the menu on the route form, a route is displayed in the text boxes of the route plan form
12	✓	On the home screen there are boxes for both membership number and password to be inputted and they are then validated before the user can log on
13	✓	There is an option for new members to click on a button which takes them to a form where they're able to enter their personal details to be added to the data store
14	✓	Relevant messages are displayed if the user attempts to logon on to their account and are unsuccessful. For example, if the membership number entered is too many digits, there is an appropriate error informing the user the number isn't valid
15	✓	When a new member is added via the new member form, all the data they enter is validated before they can be added to the member's table. If at least one of the fields is invalid, there is an error message
16	✓	When all of a new member's details have been validated by the system, they are added to the members table
17	✓	Each individual staff member is able to have their own login number and password and they can log on in a similar manner to a club member but they are directed to a personnel control form. On this form, they are able to add a new member, delete a member from the table, search for a member using their membership number of change a members details
18	✓	On the form for purchasing tickets, the member can select the quantity of each different ticket type they want to purchase, the system then calculates the total cost of their purchase
19	✓	When a member has selected their ticket quantities, a PayPal page is opened where they can purchase their tickets

Objective Number	Objective met?	Comments
20	-	A ticket confirmation is given but it is given even if the member hasn't finished completing their purchase on PayPal
21	-	There is a form where members can write e-mail messages and they are validated to be sent but they do not send to the test account so a club account cannot be created at present
22	-	An e-mail confirmation appears because the system recognises a message is being sent but it doesn't appear in the inbox of the test account
23	✓	On the route form, there is a dropdown menu with all the teams from Ryman youth available to select by the member
24	-	A route is displayed with the team name and distances for each part of the journey although the right directions are not always read in from the directions text file

User Feedback:

Burgess Hill Football Club Membership System Client Feedback Form

Positives of System:

+ The layout is very clear
+ It is easy to navigate around
+ The system looks professional and relevant to the club
+ There is a good level of security for the system
+ The staff area is made very clear so personnel can manage members easily
+ The error messages displayed are very specific
+ There is a very simple method of selecting the tickets required by a member
+ People wishing to join the club can be easily added as a new member
+ The payment method is easy

Negatives of System:

- It is not clearly labelled that the OAP tickets are also £5 like members fee
- The e-mail function doesn't work so there is still no way of setting up a club account for members to e-mail
- There is no way for extra staff member logons to be added
- When a route is displayed, the postcode of the away club is not shown
- The ticket confirmation form is displayed on clicking the pay button rather than after the transaction has been processed
- The print button on the ticket confirmation serves no purpose

Analysing User Feedback:

The user feedback suggests that the majority of the system design

Potential Future Developments:

After analysing the feedback from the user and evaluating the system I created, I have come up with a list of things that would help the system to develop in the future. These include improving the security of the system, constructing it in a more effective manner and extending the functionality of some parts

- In the future, I would change the algorithm that I have used to encrypt the passwords of the club members. The substitution cipher that was implemented was not particularly secure and an asymmetric encryption could be implemented in the future to ensure the members account is as protected as it possibly can be
- The route planner could also be extended to include teams from Burgess Hill's first team's league and the women's team's league to provide directions over larger distances
- At present, when a new member joins, the password they enter on the new member confirmation screen isn't validated at all meaning anything can be entered in the text box and added to the member's record. In the future, this can be validated to ensure it is an appropriate length and contains a combination of numbers and letters
- At the moment, the system shows the password when the user is logging on. In the future there could be stronger password security by covering the visible characters with asterisks when the user is on the home screen
- A negative point about the system is that the PayPal payment is not very well integrated because the code reads a HTML file which directs the user to the external webpage to pay for their tickets and then returned to a confirmation form. In the future, more of the payment method could be done on the forms in the system or the page could be loaded on one of the forms to make the usability of the system better for members. Further to this, the system doesn't play a part in validating the payment details of the member
- In the future, it would be beneficial if there was an option on the personnel control form for staff to filter members by a category. For example, if the number of people who had bought members tickets needed to be queried, the personnel could select members tickets and the form would show all the people who had purchased members tickets and the quantities that they had ordered
- Currently, when a postcode is validated on the personnel control form or the new members form, only the length of the postcode is checked, it would be more efficient if each character of the postcode was check to see if it was the right format. In the future, an algorithm could be implemented in the system checking each individual character