System Maintenance

System Overview

Graphical User Interface

This part of the system provides the user with an interactable display, making the system more user friendly. It also gives the system a more commercial feel and look. It contains navigational methods via push buttons, making the system simplistic. Input components such as line edits and drop down boxes are included within the interface allowing the user to commit and amend data in the database through the system. The interface displays appropriate labels to allow the user to understand the data that is being displayed.

Home Screen

This area of the system provides the user with a main menu for which they can use to navigate around the program. Each part of the program is selectable via a clearly labelled push button. The user will go through this screen every time they wish to switch areas on the system.

Squad List

This area of the system provides the user with a list of all the players they have previously entered. Each player is selectable if the user wishes to view/amend their details. It also gives the user the option to add a new player or return to the home screen.

Goal List

This area of the system provides the user with a table of all the data on goals that they have previously entered. The data is displayed under clear headers so that it is easy to read and interpret. The user is also given the option to add a new goal or return to the home screen.

Match List

This area of the system provides the user with a table of all the data on matches that they have previously entered. The data is displayed under clear headers so that it is easy to read and interpret. The user is also given the option to add a new match or return to the home screen.

Player Info

This area of the system provides the user with the all the details on a previously selected player. The data is displayed in line edits so that it can be edited. Each line edit is labelled so that the user knows what data is required in each line edit. At the bottom of the screen the user is given the option to save their changes, delete the selected player from the system or cancel (return to the previous screen).

Add Player

This area of the system provides the user with a blank detail screen, the user is prompted to fill in all the line edits with the details required to add a new player to the system. Each one of these line edits will be validated. At the bottom of the screen the user has the option to add the new player to the system and the option to cancel (return to the previous screen).

Add Goal

This area of the system provides the user with two drop down boxes and a line edit, the user is prompted to select an option from the drop down boxes and fill in the line edit with the details required to add a new goal to the system. The line edit will be validated. At the bottom of the screen the user has the option to add the new goal to the system and the option to cancel (return to the previous screen).

Add Match

This area of the system provides the user with a blank detail screen, the user is prompted to fill in all the line edits with the details required to add a new match to the system. Each one of these line edits will be validated. At the bottom of the screen the user has the option to add the new match to the system and the option to cancel (return to the previous screen).

Team Sheet

This area of the system provides the user with a formation to which they can add players to lay out a team for any upcoming matches. Each position is a push button that is clearly labelled, so that it is easy for the user to add a player to the team sheet in a certain position. When a player is entered the position remains a push button so that the player can be swapped. At the bottom of the screen the user has the option to clear the team sheet of all players or return to the home screen.

Player Select

This area of the system provides the user with a list of selectable players (name only), same layout as the squad list, however there is only options to add the selected entry to the team sheet or cancel and return to the team sheet.

Code Structure

Squad List

1. **from** PyQt4.QtGui **import** \*
2. **from** PyQt4.QtCore **import** \*
4. **from** PlayerDatabase **import** \*
5. **from** errorMessage **import** \*
6. **class** AddPlayer(QMainWindow):
7. **def** \_\_init\_\_(self, parent):
8. super().\_\_init\_\_(parent)
10. self.parent = parent
12. self.setWindowTitle("Add Player")
13. *#Widget setting*
14. self.forename = QLineEdit()
15. self.labelF = QLabel("Forename: ")
17. self.surname = QLineEdit()
18. self.labelS = QLabel("Surname: ")
20. self.rating = QLineEdit()
21. self.labelR = QLabel("Rating: ")
23. self.email = QLineEdit()
24. self.labelE = QLabel("Email: ")
26. self.position = QLineEdit()
27. self.labelP = QLabel("Position: ")
29. self.avaliable = QLineEdit()
30. self.labelA = QLabel("Avaliable: ")
32. self.btnAdd = QPushButton("Add")
33. self.btnCancel = QPushButton("Cancel")
35. self.hlayout1 = QHBoxLayout()
36. self.vlayout1 = QVBoxLayout()
37. self.vlayout2 = QVBoxLayout()
38. self.vlayout3 = QVBoxLayout()
39. self.hlayout2 = QHBoxLayout()
40. self.vlayout1.addWidget(self.labelF)
41. self.vlayout2.addWidget(self.forename)
42. self.vlayout1.addWidget(self.labelS)
43. self.vlayout2.addWidget(self.surname)
44. self.vlayout1.addWidget(self.labelR)
45. self.vlayout2.addWidget(self.rating)
46. self.vlayout1.addWidget(self.labelE)
47. self.vlayout2.addWidget(self.email)
48. self.vlayout1.addWidget(self.labelP)
49. self.vlayout2.addWidget(self.position)
50. self.vlayout1.addWidget(self.labelA)
51. self.vlayout2.addWidget(self.avaliable)
52. self.vlayout1.addWidget(self.btnAdd)
53. self.vlayout2.addWidget(self.btnCancel)
54. self.hlayout1.addLayout(self.vlayout1)
55. self.hlayout1.addLayout(self.vlayout2)
56. self.vlayout3.addLayout(self.hlayout1)
57. self.hlayout2.addWidget(self.btnAdd)
58. self.hlayout2.addWidget(self.btnCancel)
59. self.vlayout3.addLayout(self.hlayout2)
60. self.widget = QWidget()
61. self.widget.setLayout(self.vlayout3)
62. self.setCentralWidget(self.widget)
64. self.btnAdd.clicked.connect(self.btnAdd\_pushed)
65. self.btnCancel.clicked.connect(self.btnCancel\_pushed)

68. **def** btnAdd\_pushed(self):
69. *#Assigning 'forename' to the inputted forename as text*
70. forename = self.forename.text()
71. *#Assigning 'surname' to the inputted surname as text*
72. surname = self.surname.text()
73. *#Assigning 'rating' to the inputted rating as text*
74. rating = self.rating.text()
75. *#Assigning 'email' to the inputted email as text*
76. email = self.email.text()
77. *#Assigning 'position' to the inputted position as text*
78. position = self.position.text()
79. *#Assigning avaliable to the inputted avaliable  as text*
80. avaliable = self.avaliable.text()
82. *#Validation*
83. email\_valid = False
84. rating\_valid = False
85. position\_valid = False
86. avaliable\_valid = False
87. forename\_valid = False
88. surname\_valid = False
89. *#Checking that each field has an entry of some sort*
90. **if** forename != "" **and** surname != "" **and** rating!= "" **and** email != "" **and** position != "" **and** avaliable != "":
91. *#Setting up alphabet lists*
92. alphabet\_lower = []
93. alphabet\_upper =[]
94. *#Adding each lower case letter to the lower list*
95. **for** letter **in** map(chr, range(97, 123)):
96. alphabet\_lower.append(letter)
97. *#Adding each upper case letter to the upper list*
98. **for** letter **in** map(chr, range(65,91)):
99. alphabet\_upper.append(letter)
100. *#Forename validation*
101. forename\_valid = True
102. count = -1
103. **for** each **in** forename:
104. count = count +1
105. **if** count == 0:
106. **if** forename[count] **not** **in** alphabet\_upper:
107. forename\_valid = False
108. **else**:
109. **if** forename\_valid == True **and** each **not** **in** alphabet\_lower:
110. forename\_valid = False
112. **if** forename\_valid == False:
113. *#Running the Error widget, passing in the message to display*
114. self.error = ErrorWindow(self,"Please enter a valid Forename")
115. *#Showing the error window*
116. self.error.show()
117. *#Raising the error window to the front of the screen*
118. self.error.raise\_()
120. *#Surname validation*
121. surname\_valid = True
122. count = -1
123. **for** each **in** surname:
124. count = count +1
125. *#Checking the first letter*
126. **if** count == 0:
127. *#Checking its a capital letter*
128. **if** surname[count] **not** **in** alphabet\_upper:
129. surname\_valid = False
130. **else**:
131. **if** surname\_valid == True **and** each **not** **in** alphabet\_lower:
132. surname\_valid = False
134. **if** surname\_valid == False:
135. *#Running the Error widget, passing in the message to display*
136. self.error = ErrorWindow(self,"Please enter a valid Surname")
137. *#Showing the error window*
138. self.error.show()
139. *#Raising the error window to the front of the screen*
140. self.error.raise\_()
142. *#Email Validation*
143. at\_valid = False
144. dot\_valid = False
145. *#Checking the email contains an @ and . symbol*
146. **for** each **in** email:
147. **if** each == "@":
148. at\_valid = True
149. **if** each == ".":
150. dot\_valid = True
151. *#Setting email\_valid to true if the input contains an @ and an .*
152. **if** at\_valid == True **and** dot\_valid == True:
153. email\_valid = True
155. **else**:
156. *#Running the Error widget, passing in the message to display*
157. self.error = ErrorWindow(self,"Please enter a valid Email address")
158. *#Showing the error window*
159. self.error.show()
160. *#Raising the error window to the front of the screen*
161. self.error.raise\_()

164. *#Rating validation*
165. *#List of valid entries*
166. **if** rating **in** ["0","1","2","3","4","5","6","7","8","9","10"]:
167. rating\_valid = True
168. **else**:
169. *#Running the Error widget, passing in the message to display*
170. self.error = ErrorWindow(self,"Please enter a valid Rating")
171. *#Showing the error window*
172. self.error.show()
173. *#Raising the error window to the front of the screen*
174. self.error.raise\_()
176. *#Position Validation*
177. *#List of valid entries*
178. **if** position **in** ["GK","LB","CB","RB","LM","CM","RM","ST"]:
179. position\_valid = True
180. **else**:
181. *#Running the Error widget, passing in the message to display*
182. self.error = ErrorWindow(self,"Please enter a valid Position")
183. *#Showing the error window*
184. self.error.show()
185. *#Raising the error window to the front of the screen*
186. self.error.raise\_()
188. *#Avaliable Validation*
189. *#List of valid entries*
190. **if** avaliable **in** ["Y","y","YES","Yes","N","n","NO","no"]:
191. avaliable\_valid = True
192. **else**:
193. *#Running the Error widget, passing in the message to display*
194. self.error = ErrorWindow(self,"Please enter a valid avaliabilty")
195. *#Showing the error window*
196. self.error.show()
197. *#Raising the error window to the front of the screen*
198. self.error.raise\_()
200. **else**:
201. *#Running the Error widget, passing in the message to display*
202. self.error = ErrorWindow(self,"Please enter data into all the required fields")
203. *#Showing the error window*
204. self.error.show()
205. *#Raising the error window to the front of the screen*
206. self.error.raise\_()
208. **if** email\_valid == True **and** rating\_valid == True **and** position\_valid == True **and** avaliable\_valid == True **and** forename\_valid == True **and**surname\_valid == True:
209. g\_database.AddPlayer(forename, surname, int(rating), email, position, avaliable)
210. self.parent.show()
211. self.parent.refresh\_List()
212. self.close()


216. **def** btnCancel\_pushed(self):
217. self.parent.show()
218. self.close()