**Module: 206CDE Real World Project – Component 2**

Tutorial Session: 4

Group ID: 1

Tutor Name: Nazaraf Shah

Group Name: The Dictators

Software Solution: Moral Dictator

Members: Dawid Lominski, William Read, Peter Kirby, Adrian Putra, Joshua Pitched and Reece Robinson

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Student ID** | **Mobile** | **Email** |
| William Read | 6230409 | 07548940943 | readw@coventry.ac.uk |
| Dawid Lominski | 6108614 | 07598884269 | lominskd@uni.coventry.ac.uk |
| Peter Kirby | 6317526 | 07715815226 | allinone95@hotmail.com |
| Reece Robinson | 5598960 | 07463800232 | djstutz@outlook.com |
| Adrian Putra | 6428615 | 07487752106 | putraa@uni.coventry.ac.uk |
| Joshua Pitcher | 6384779 | 07860137580 | pitcher6@uni.coventry.ac.uk |

**What is the problem we tried to address?**

Right now, there are gaps in the market when it comes to understanding people’s personalities using a game, at the same time making it enjoyable but also suitable for a research on how people choose the decisions that they do and what can their reasoning be behind it.

**The proposed solution.**

A simple game where you get a choice of being a dictator or a party and as a player you get to choose which one you want to play as. You are faced with different moral choices that will have an impact on your civilization but also might have an effect on other civilizations. The game is going to use machine learning for responsive difficulty, meaning that the data is going to be collected on how well the player is doing and adjust the difficulty when new game is started, and during the gameplay, which will make it more challenging for the user. All the results will be updated to the database, analysed and displayed on our website to show how other players are comparing with each other.

**Issues**

* The less users playing our game, the lower the performance there will be, this is because when comparing users against each other, the higher the sample pool, the more accurate the results will be,
* Although we will have a large pool of questions, sometimes they might repeat, this is because the questions asked will be randomly chosen from the database,
* We tried our best when writing the questions to make sure that they will not be offensive and will not harm anyone in anyway possible, but we can’t promise that they won’t make people sad about certain things. In the future the user will have an option to report the questions if something like this happens.

**Potential competitors**

* [Morality Play (http://www.philosophyexperiments.com/moralityplay/](Morality%20Play%20(http://www.philosophyexperiments.com/moralityplay/)) – one of the competitors asks the user 19 questions, and at the end they provide the analysis on the answers and a description on what the score means. They use moral parsimony to display results to users,
* Moral Dilemma (<http://moraldilemmathegame.com/>) – another competitor is a card based game which the user actually have to buy. There is no professional analysis provided at the end of the game, and it is just mainly a game to play with your friends.
* Scruples (<http://scruplesgame.com/>) - this is a game that you can play with your friends on an iPhone, it has a pool of 225 questions, and it was basically made for getting to know each other. There is no analysis at the end.
* Moral Machine (<http://moralmachine.mit.edu/>) – This is most likely one of our largest competitors. It has been made by MIT, and it asks questions about how an autonomous car should behave in certain situations. At the end it provides a full analysis based on your answers, and what kind of characters are most valuable to you.

**Machine Learning**

Type of artificial intelligence that provides computers with the ability to learn without being explicitly programmed. Development on computer programs that can change when exposed to new data. Similarly to data mining, machine learning goes through the data provided and looks for patterns, but instead of extracting the data for human comprehension, it uses the data to find patterns and adapt program actions accordingly, without necessarily programming everything. (whatis)

Supervised machine learning

Majority of machine learning uses supervised learning. The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (y) for that data. (machinelearningmastery)

Called supervised learning because the process of an algorithm learning from the training dataset can be though of as a teach supervising the learning process. We know the correct answers, the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance.

**Problems:**

* Classification – a classification problem is when the output is a category, such as “red” or “blue” or “disease” and “no disease”.
* Regression – A regression problem is when the output variable is a real value, such as “dollars” or “weight”

**Examples**:

Linear regression for regression problems  
Random forest for classification and regression problems,  
Support vector machines for classification problems.

Unsupervised machine learning

When you only have input data but no corresponding output variables. The aim for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data.

Called unsupervised learning because unlike supervised learning there is no correct answers and there is no teacher. Algorithms are left to their own devises to discover and present the interesting structure in the data.

**Problems:**

* Clustering – where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behaviour
* Association – where you want to discover rules that describe large portions
* of your data, such as people that buy X also tend to buy Y.

**Examples:**

k-means for clustering problems,  
Apriori algorithm for association rule learning problems

Semi-supervised machine learning

Problems where you have a large amount of input data(x) and only some of the data is labelled (y) are called semi-supervise learning problems.

**Supervised –** All data is labelled and the algorithms learn to predict the output from the input data.

**Unsupervised -**  All data is unlabelled and the algorithms learn to inherent structure from the input data

**Semi-supervised** – Some data is labelled but most of it is unlabelled and a mixture of supervise and unsupervised techniques can be used.

**References:**

Rouse, M. (n.d.) *Cite A Website - Cite This For Me* [online] available from <http://whatis.techtarget.com/definition/machine-learning> [6 February 2017]

Brownlee, J. (2016) *Supervised And Unsupervised Machine Learning Algorithms - Machine Learning Mastery* [online] available from <http://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/> [6 February 2017]

**UML Diagram**



Figure 1 – an UML diagram showing all the tables in the database, their attributes and relationships between each other. Our aim was to create the most efficient database that we could, but also wanted to make sure that it wasn’t overcomplicated and that it is easy to access. This UML diagram only shows the databases for the actual game, not the website.

**USE CASE Diagram**

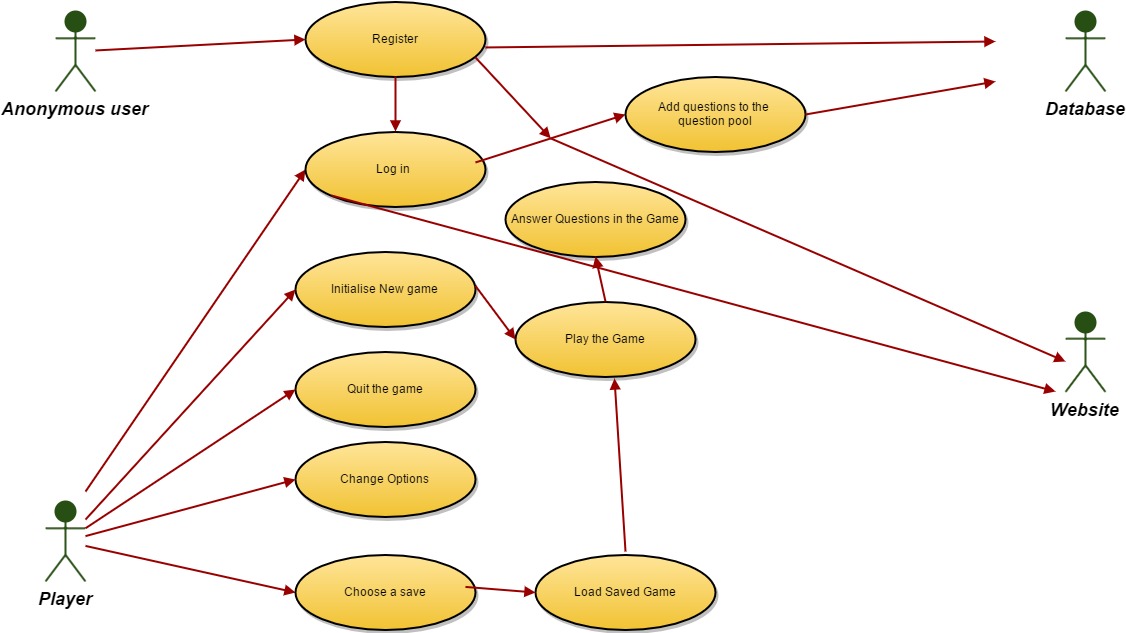
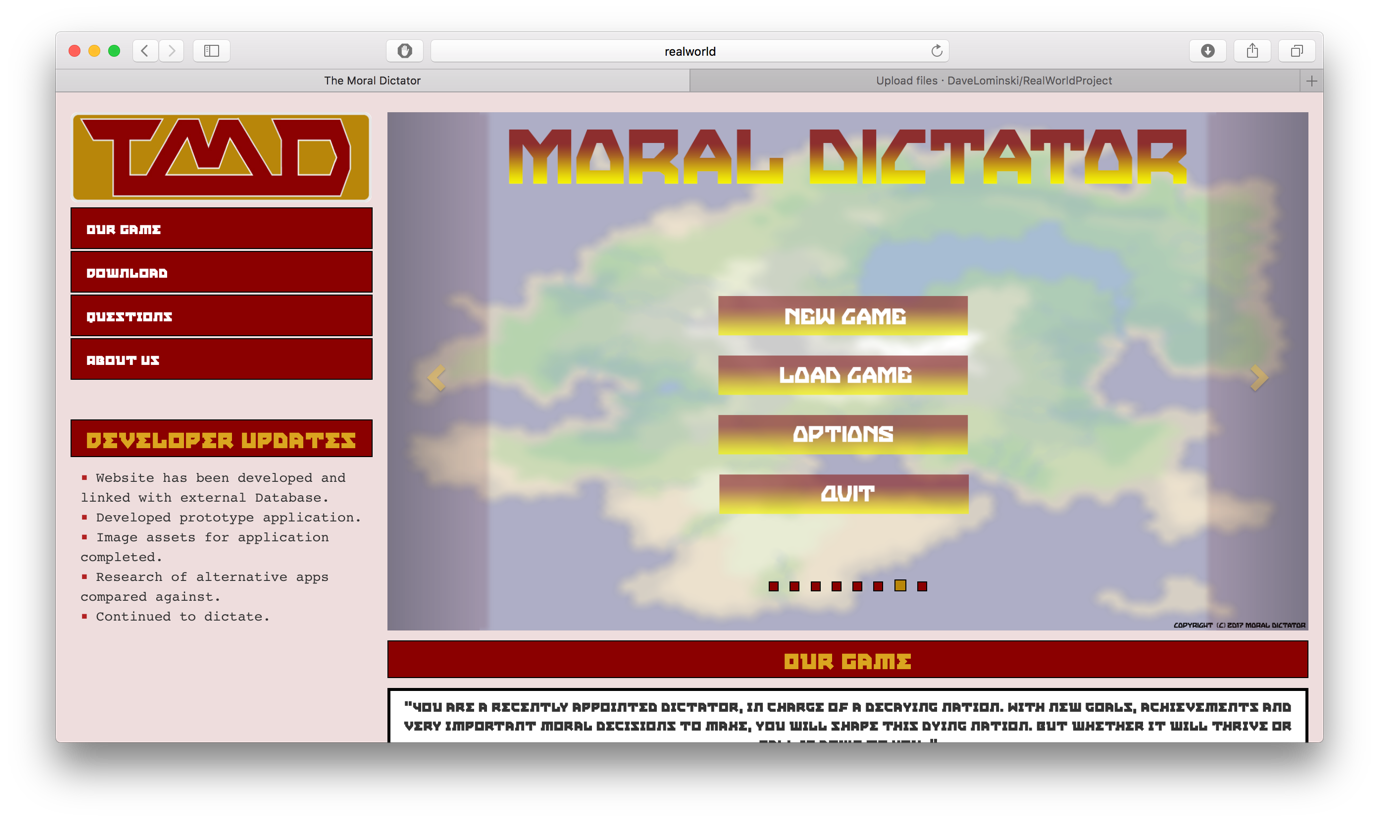


Figure 2 – A USE CASE Diagram showing how the whole software solution will work.

This is a really important diagram, as it basically shows how our software works. An anonymous user has only access to registering on the website, once the account is made the user can log in on the website and use its full functionality. Once the user is registered, all the data is sent to the database. Once player accesses the game, they will be able to initialise a new game, quit the game, change options and also choose a save.

**Web Application Screenshots**

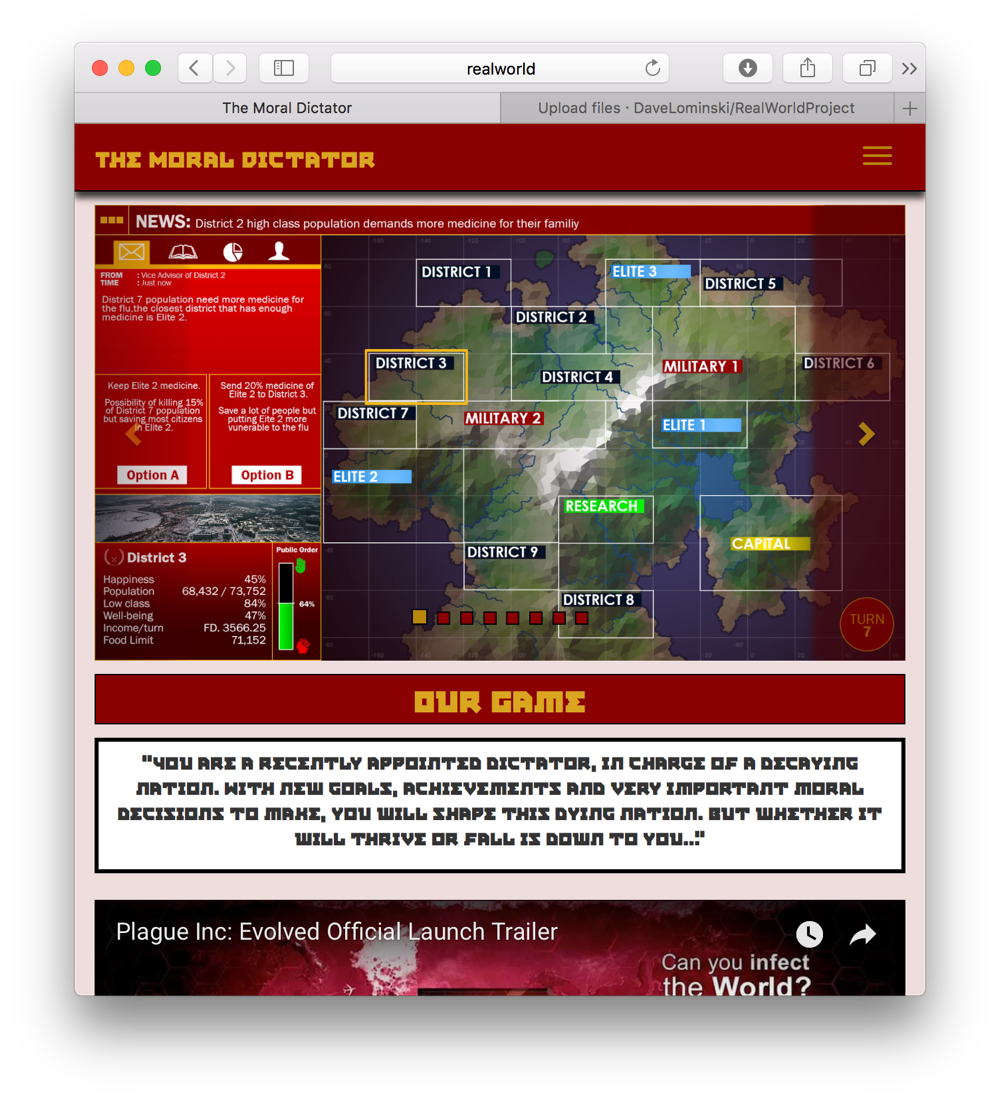
****

Figure 3 – Image to show slideshow interface of our Web application, which updates as more screenshots of the game are added to the Media/Images folder.

Figure 4 – Image showing the system being designed to fit a Mobile Friendly audience, allowing users to access and use the site from any device.

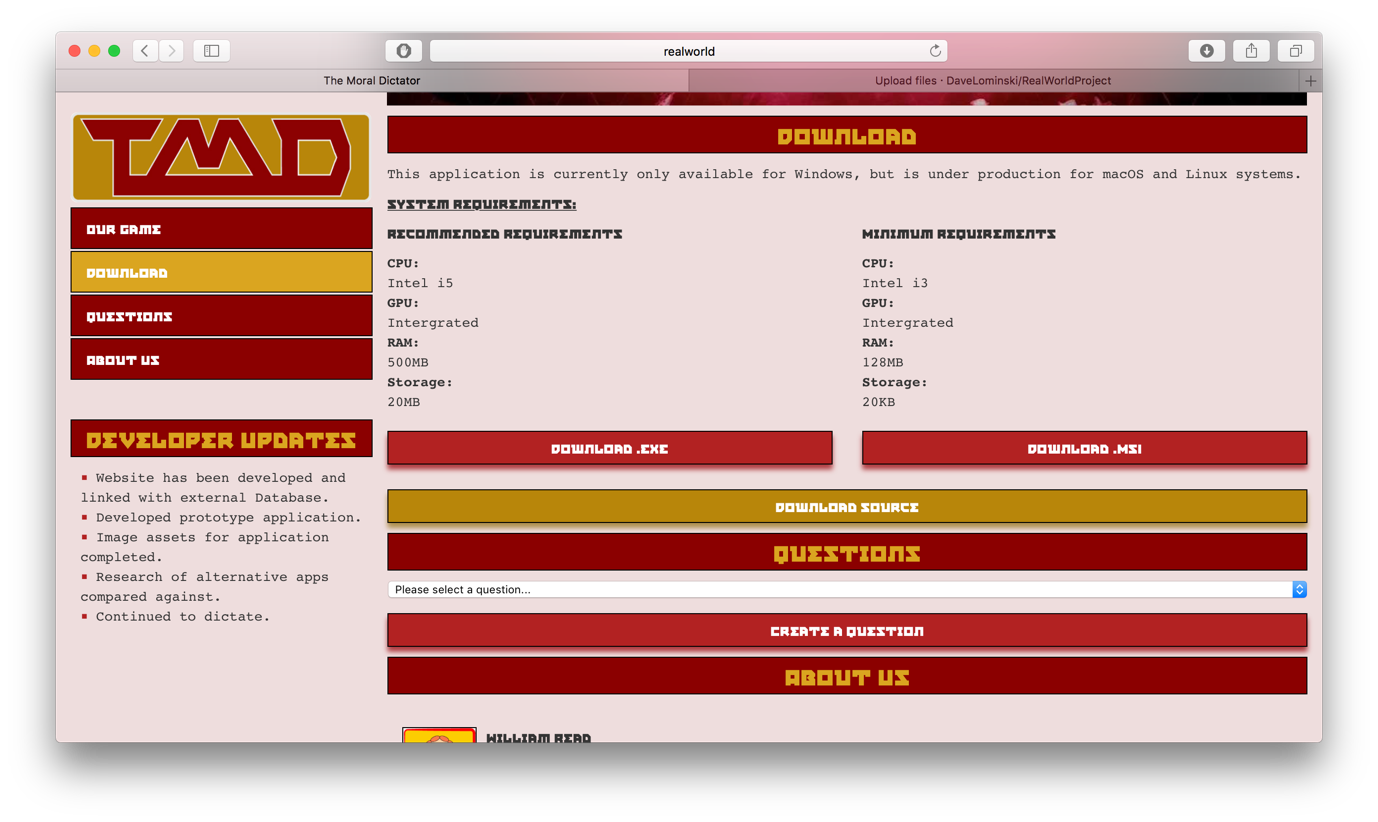


Figure 5 – Image shows the display of where a user can download their application and play the overall game, or can improve and add their own contributions via use of the source code.

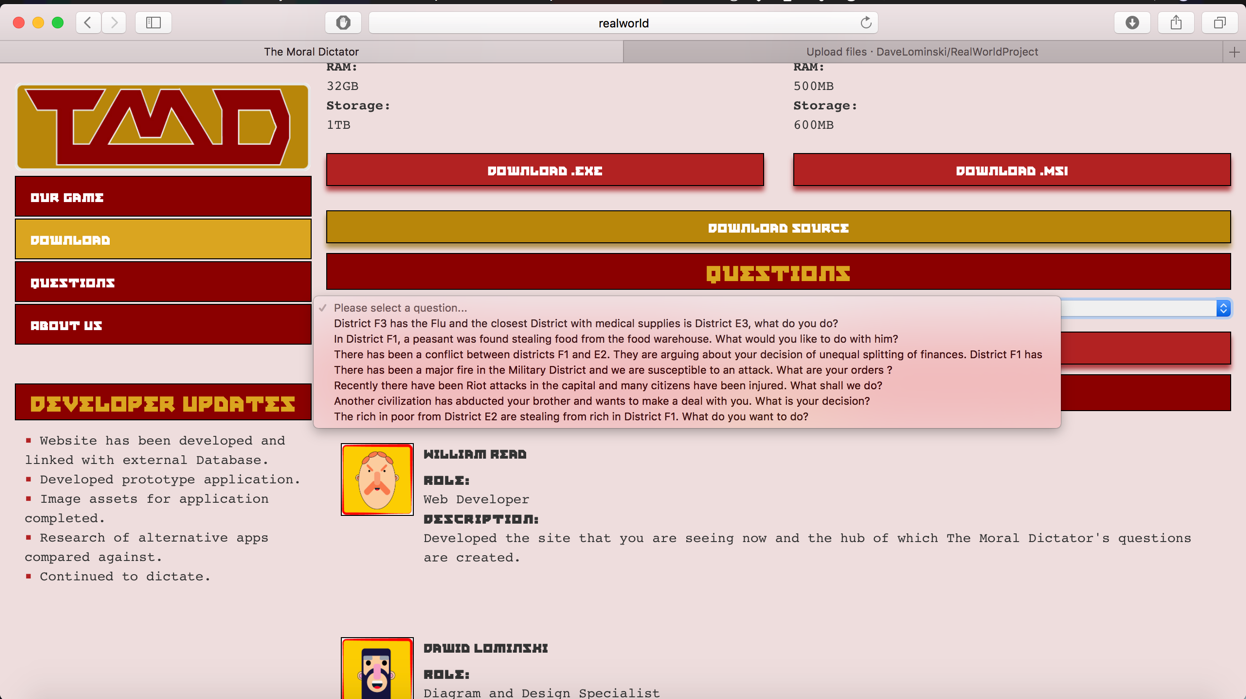
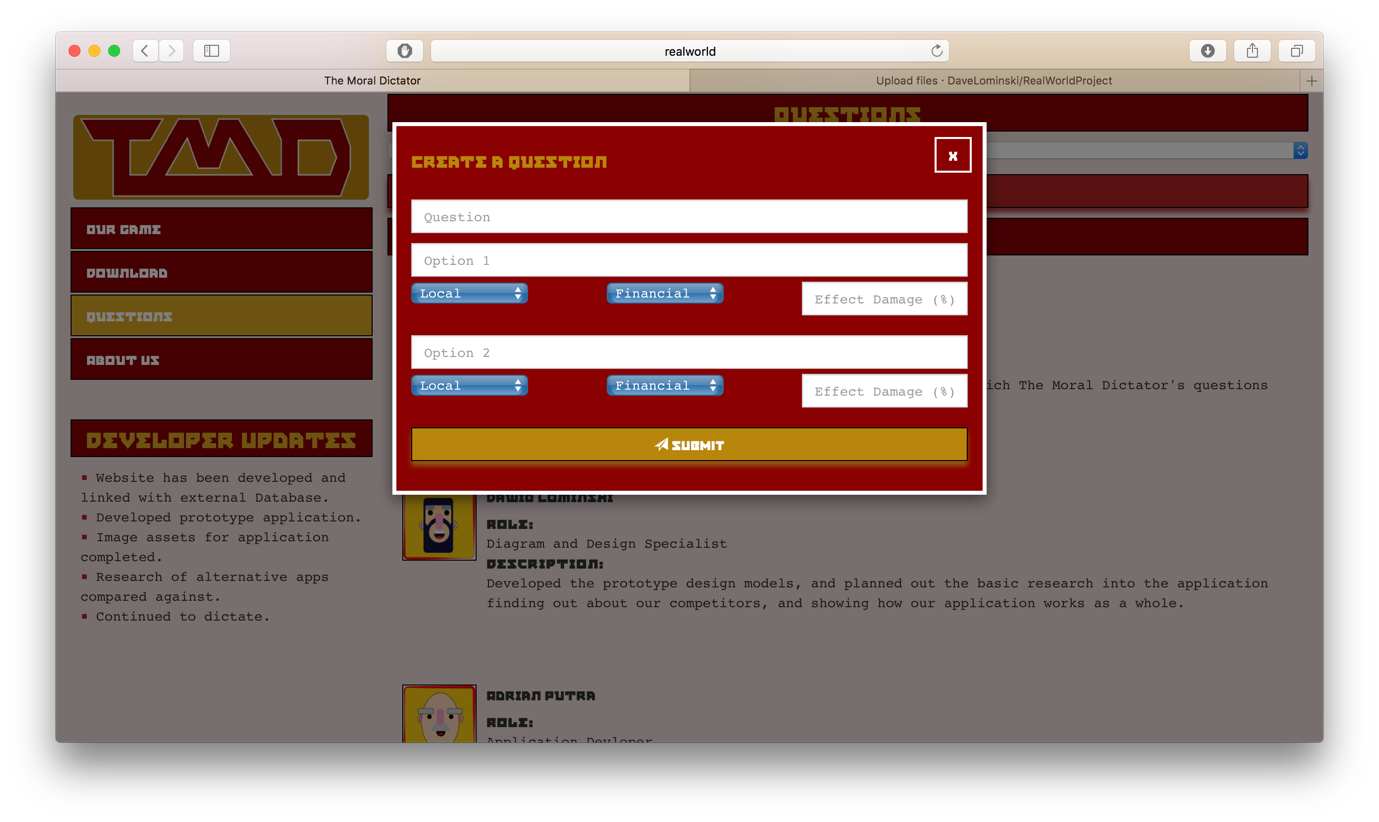


Figure 6 – Image showing that all the questions in the database are gathered, and when selected displays the information on how people have answered each of the questions.



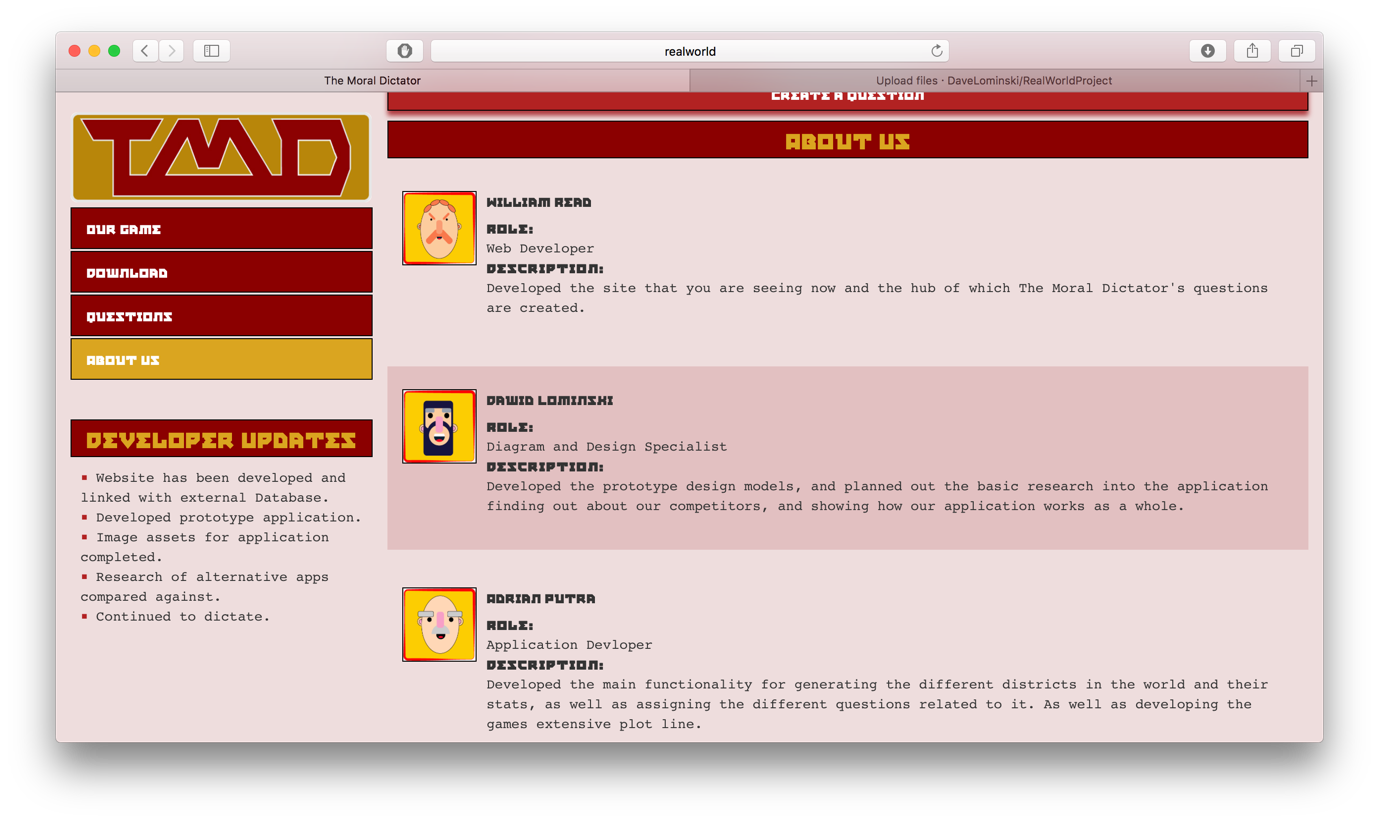


Figure 7 – Image showing the method of a user being opening the create question modal to create a question allowing the user to enter different attributes for each answer to the desired question.

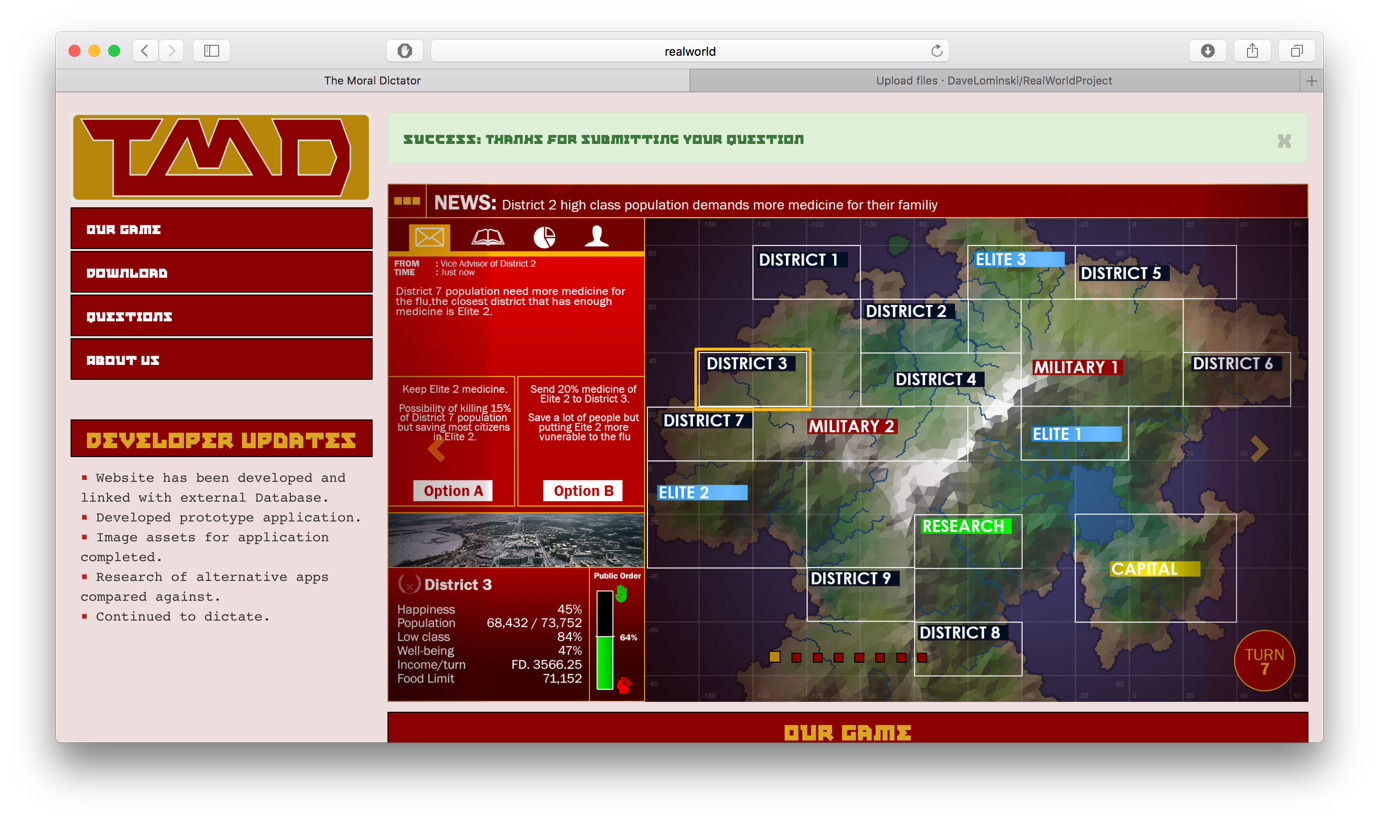


Figure 8 – Image to show the result of a Post back from the server saying that a Question submission has been successful.

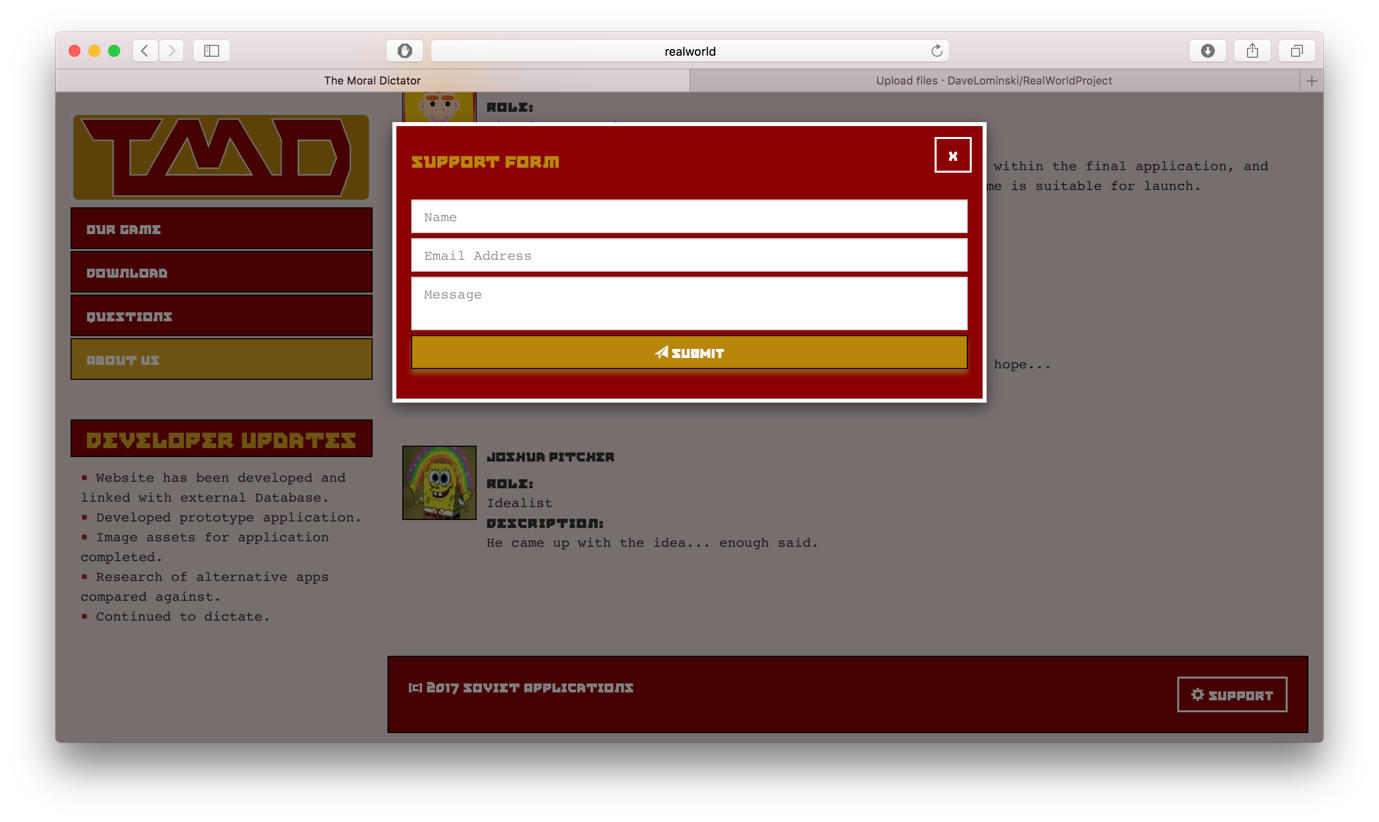
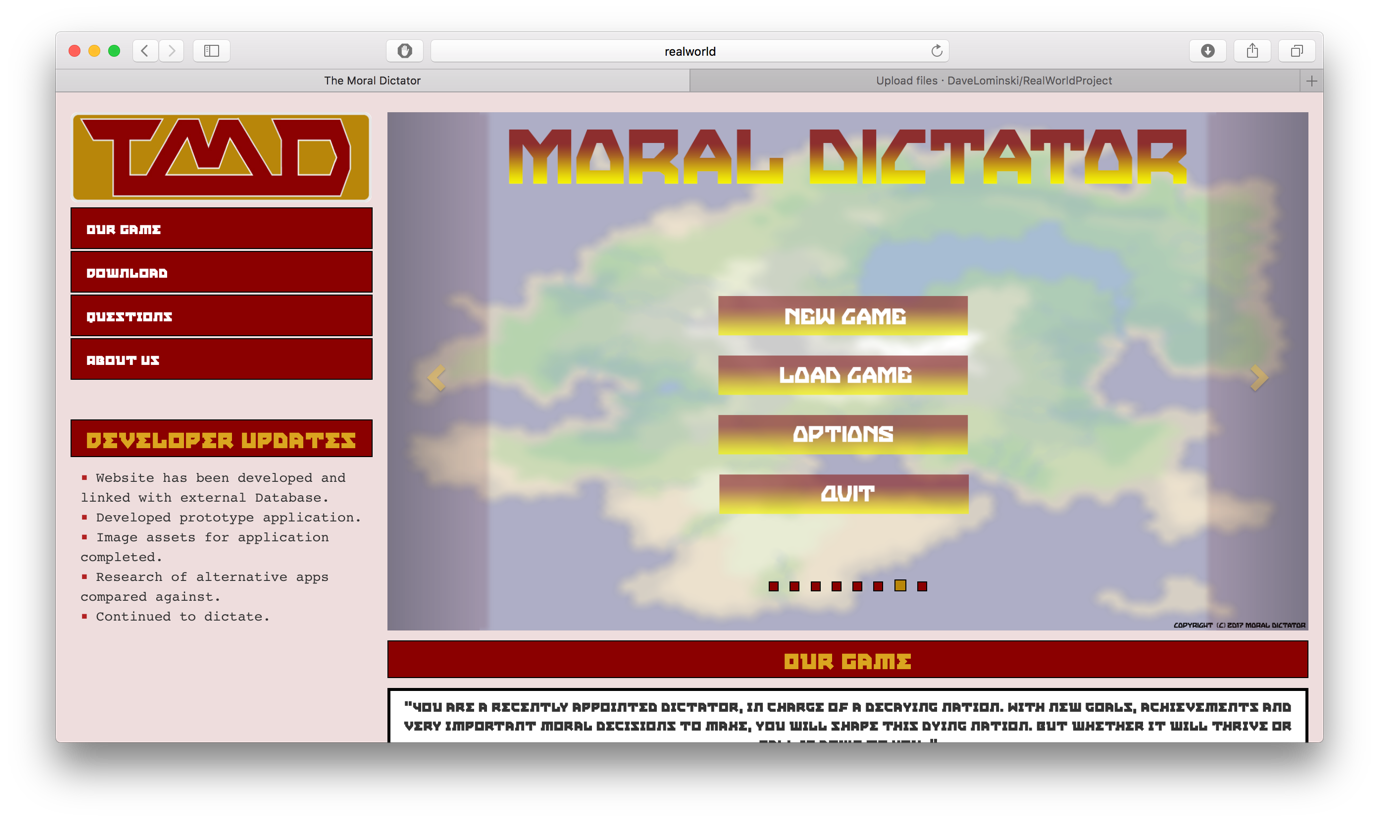
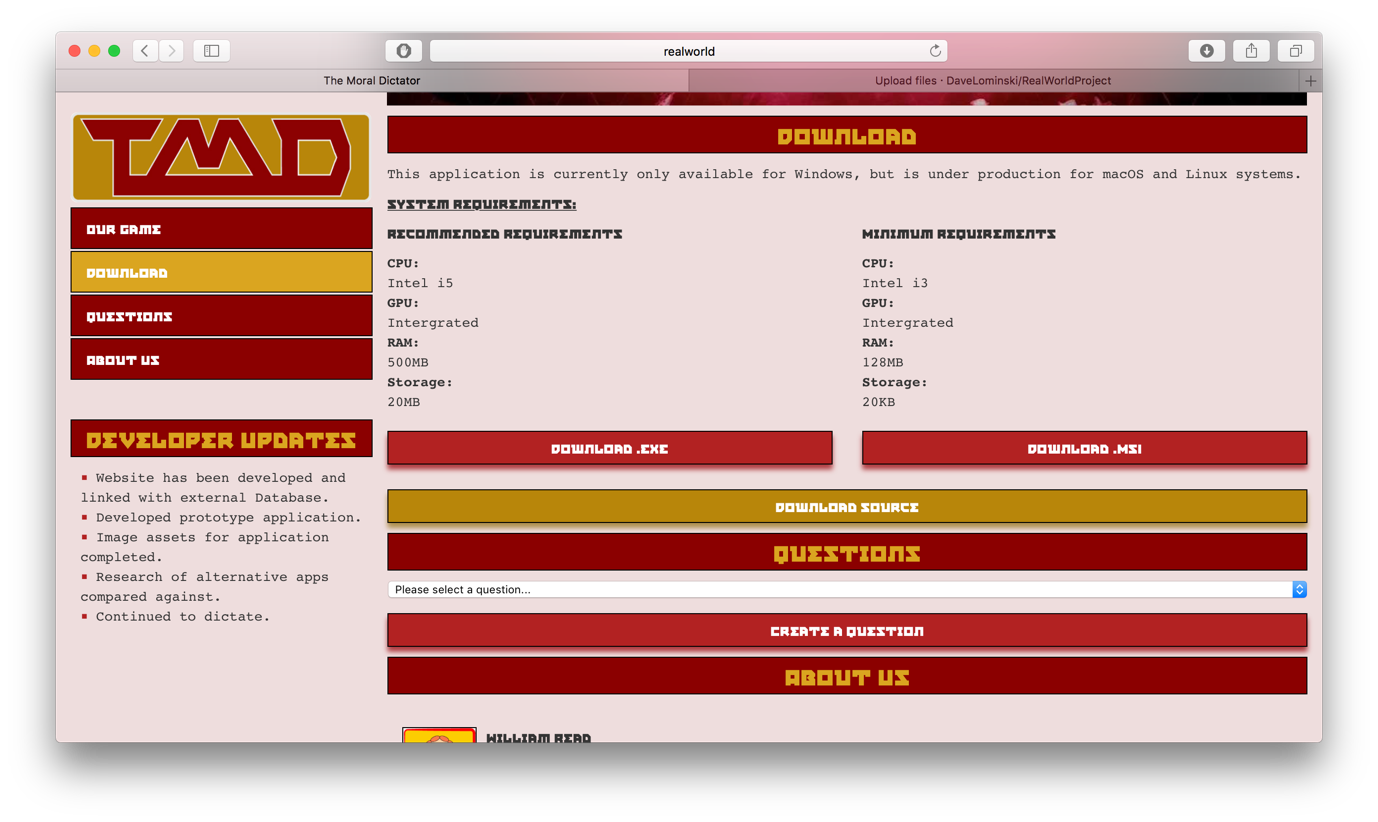
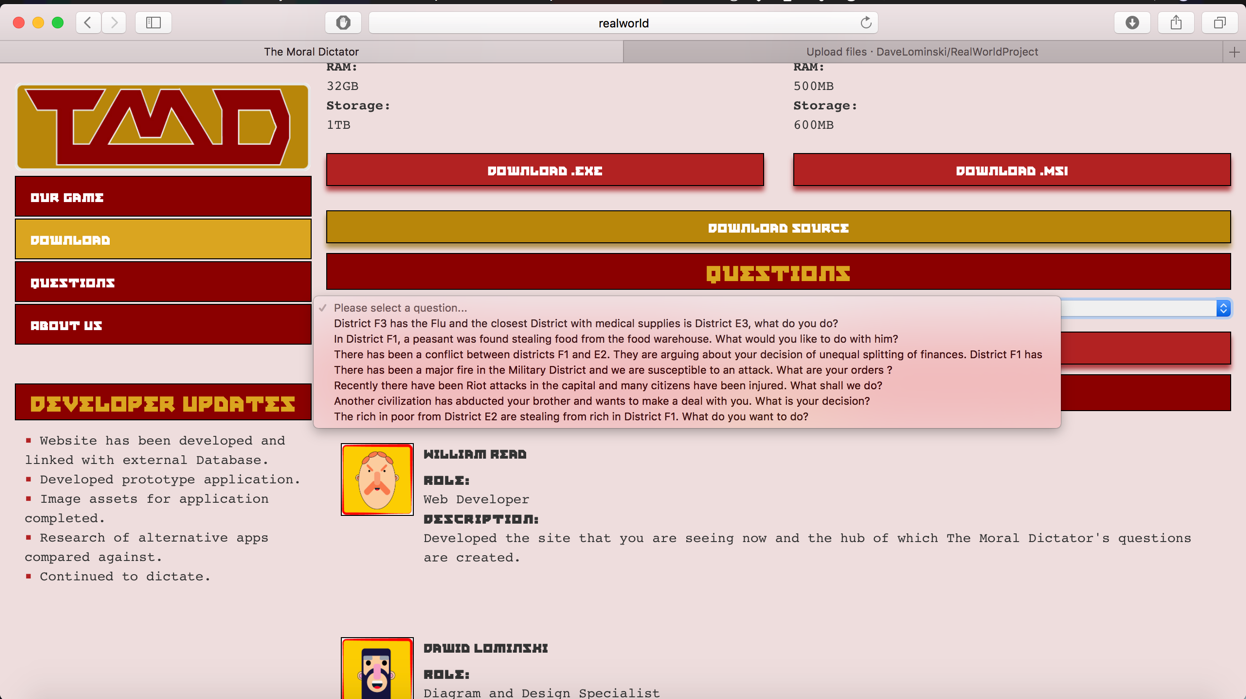
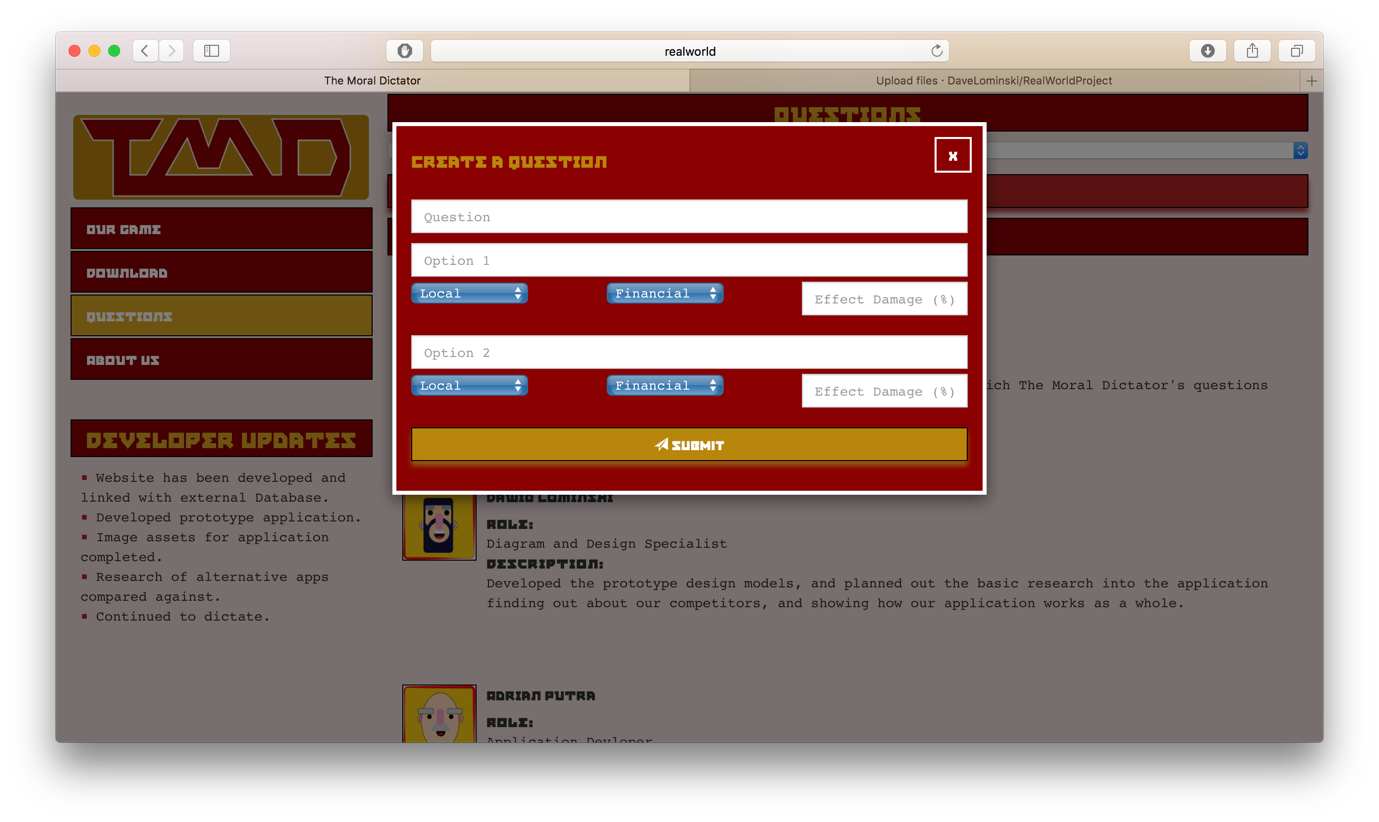
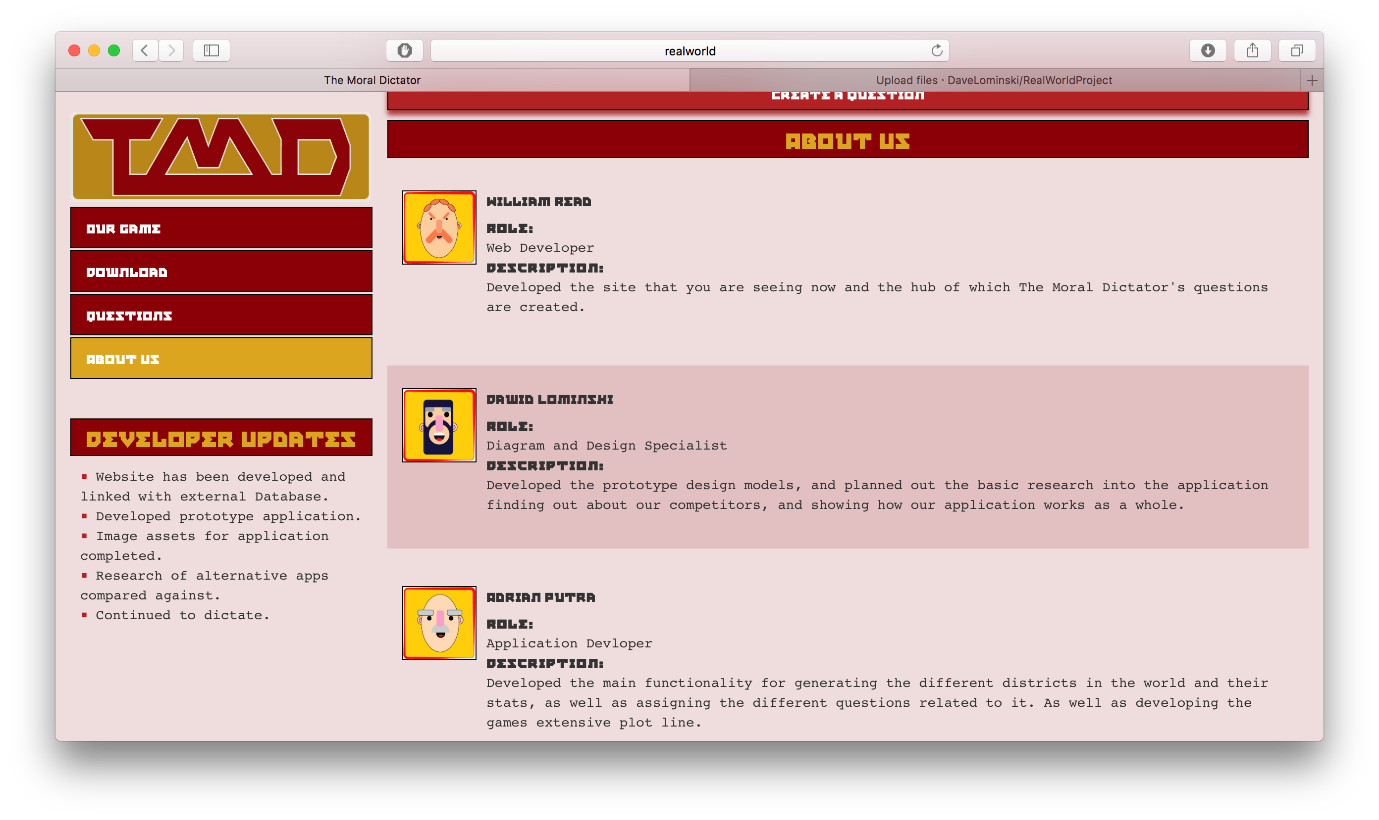


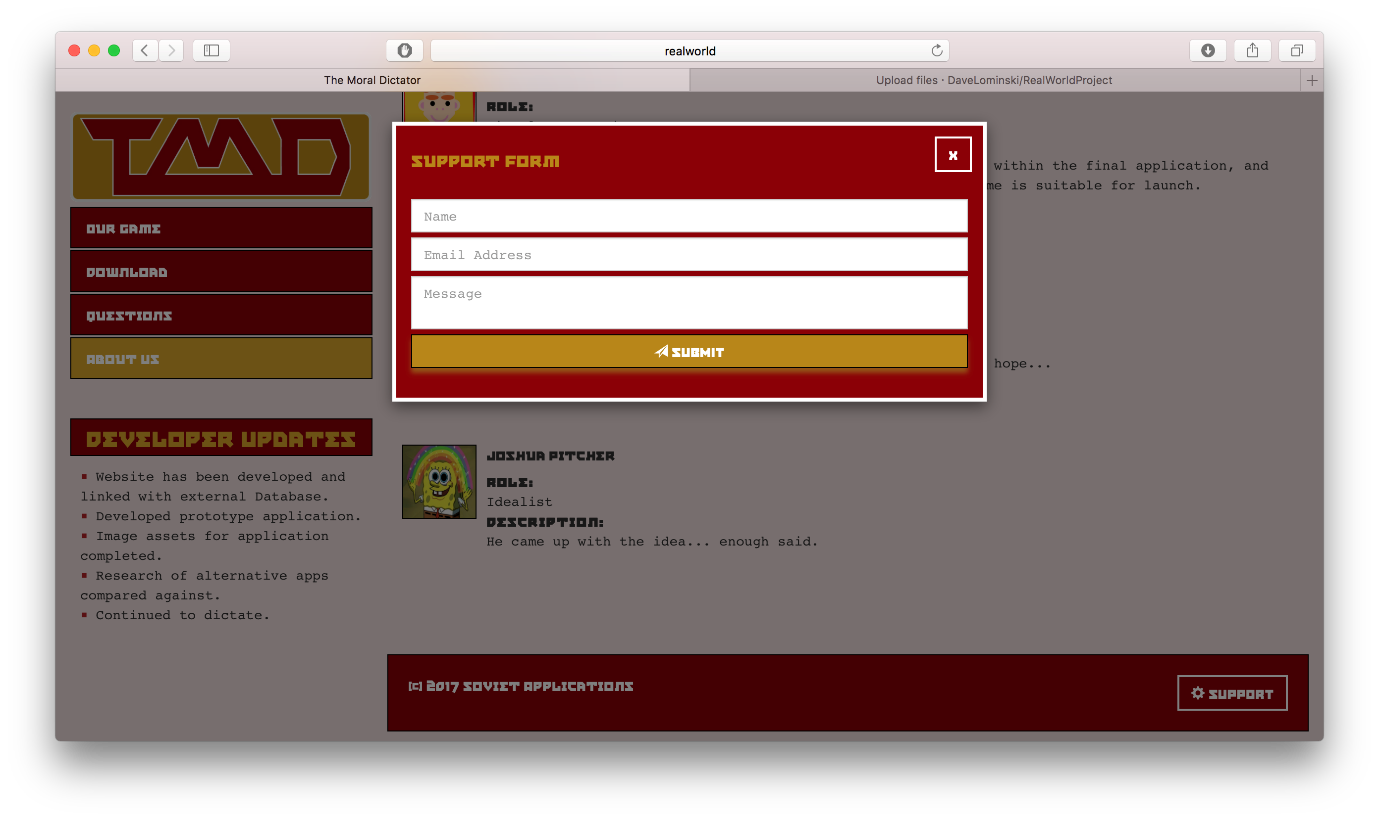
Figure 9 – Image to show the modal of which the support form is displayed.

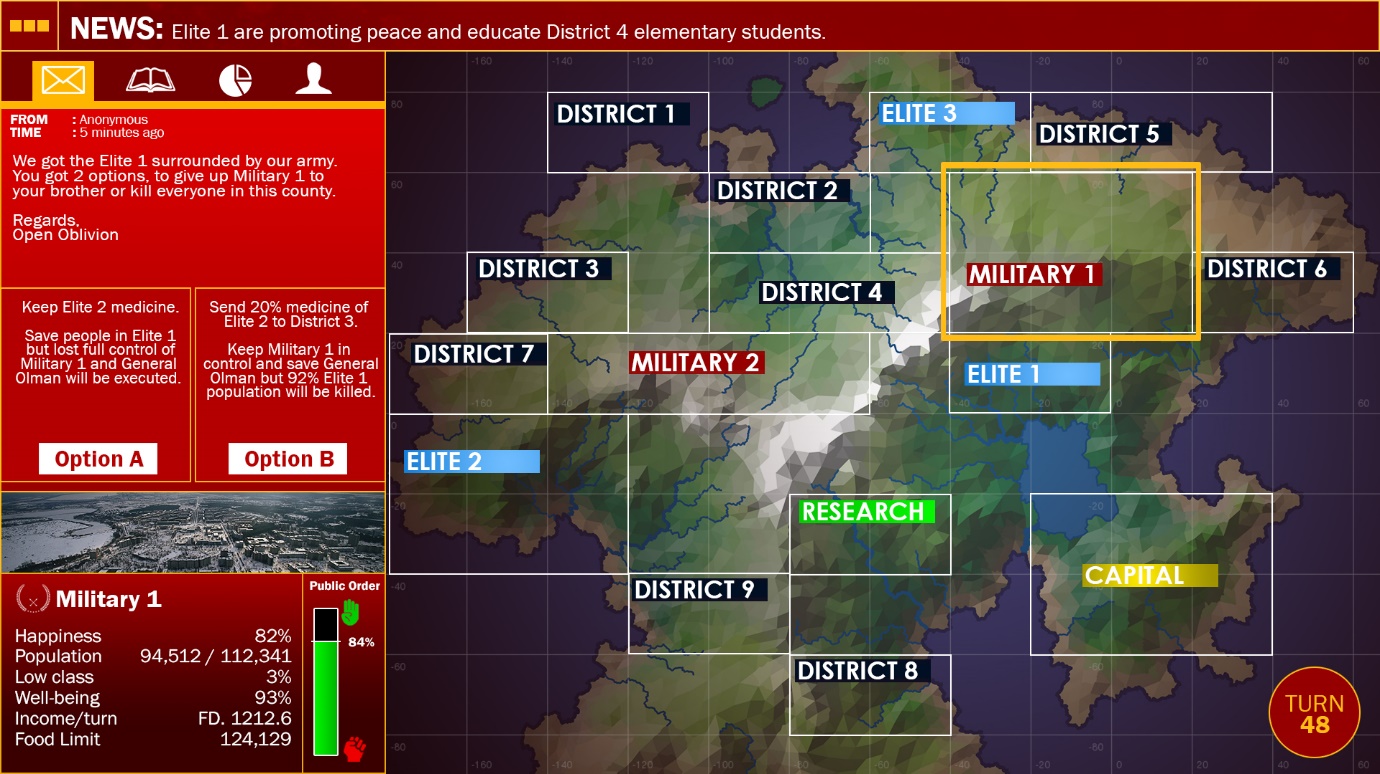
**Interfaces**

1. ****
2. 
3. 



1. 



1. 
2. 
3. 
4. 

**MORAL DICTATOR Documentation**

**Why people want to play?**

* **Turn-based Strategy game**
* **Game mode**:
  + Campaign (Maintain 60 Turns)
  + Survival (as long as possible: Easy, Normal, Hard, Brutal)
  + Never-ending (Machine learning, adjust your capability and give feedback)
  + Sandbox (customise your own game)
* **Interesting story plot for Campaign**:
  + The year is 1872 your name is Boris Buldan, you are the second son of Yuri Buldan the Supreme Leader of the Communist country the New Fregar
  + In 1902 your father got killed by a spy from an activist group called Open Oblivion. According to the country’s tradition your brother Gerald Buldan will take over.
  + Boris and Gerald has the same interest and wanted equality in the country
  + So Boris decided to make an agreement with Gerald. Boris will take over the power for 60 months and his brother Gerald will plan for the democracy takeover plan (take down all five of the main communist public figure)
    - Kerkov Olman – Military 1 High General
    - Fred Greez – Military 2 High General
    - Naidar Tai – Finance High General
    - Samgong Sung – District High General
    - Ahkani Logdan – Elite High General
  + But what Gerald didn’t know, Boris has his own way of doing things. You have 60 turns to be in control and make things in order, what action to take is all in your hands, good luck Supreme Leader.
  + ENDING 1:
  + ENDING 2:
  + ENDING 3:
  + ENDING 4:
  + ENDING 5:
* Game have different endings depending on the actions you choose
* **Free to Play** (available from our website and will be on Steam Greenlight when BETA is available)
* **Achievement System** (will be developed in the future…)
* **Contribution for Machine Learning research**

**Game Mechanics**

* **Campaign 60 turns events**

**Yellow is Story event**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | Random Event 1 | 21 |  | 41 | Random Event 26 |
| 2 | Gerald needed an access to Military 2 (Fred) | 22 | Random Event 13 | 42 | Random Event 27 |
| 3 | Random Event 2 | 23 |  | 43 | Random Event 28 |
| 4 | Random Event 3 | 24 | Random Event 14 | 44 | Military security for Gerald army |
| 5 | Random Event 4 | 25 | Random Event 15 | 45 | Gerald done setting up the plan |
| 6 | Open Oblivion founded in District 7 | 26 |  | 46 | Random Event 29 |
| 7 | Random Event 5 | 27 | Random Event 16 | 47 | Random Event 30 |
| 8 | Random Event 6 | 28 | Random Event 17 | 48 | Assassination of Kerkov (1st public figure) |
| 9 | Random Event 7 | 29 | Random Event 18 | 49 | Random Event 31 |
| 10 |  | 30 |  | 50 | Random Event 32 |
| 11 |  | 31 |  | 51 | Assassination of Fred (2nd public figure) |
| 12 |  | 32 | Random Event 19 | 52 | Assassination of Naidar (3rd public figure) |
| 13 | Random Event 8 | 33 | Random Event 20 | 53 | Random Event 33 |
| 14 |  | 34 | Random Event 21 | 54 | Random Event 34 |
| 15 |  | 35 | Random Event 22 | 55 | Assassination of Samgong (1st public figure) |
| 16 | Random Event 9 | 36 | New activist called KeyConscious founded by your brother | 56 | Random Event 35 |
| 17 | Gerald contracted Open Oblivion and needed hand | 37 | Random Event 23 | 57 | Random Event 36 |
| 18 | Random Event 10 | 38 | Random Event 24 | 58 | Assassination of Ahkani (1st public figure) |
| 19 | Random Event 11 | 39 | Random Event 25 | 59 | Random Event 37 |
| 20 | Random Event 12 | 40 | Assassination of KeyConscious leader | 60 | Open Oblivion come, either your brother, your son, or you died |

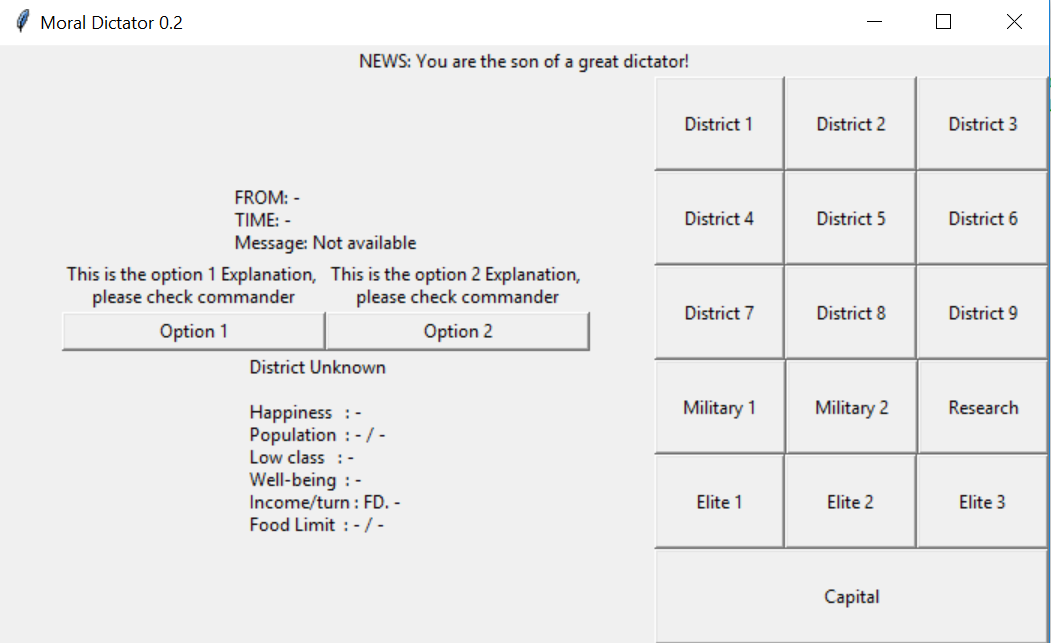
**Code:** This is a Python Code for our prototype, we have used Tkinter for the UI design. We have used Object Oriented Programming technique to make the program more efficient and run more smoothly.

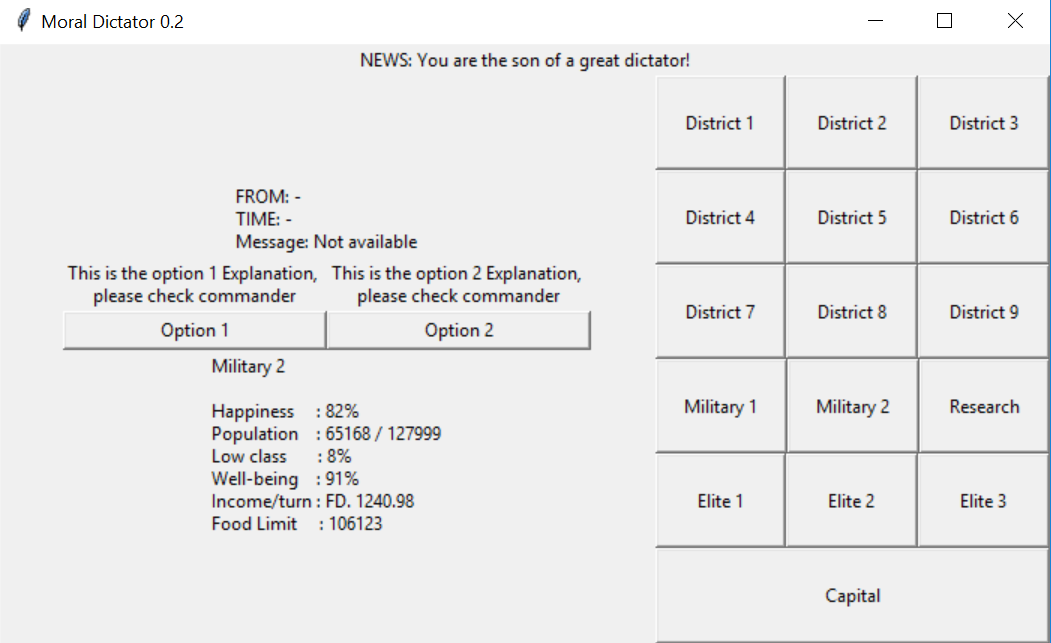
1. **from** tkinter **import** \*
2. **from** random **import** randint
3. **from** random **import** uniform
5. """ You are the son of a great dictator!
6. You have 9 town districts, 2 military base, 1 research facility, 3 elite districts, and your main capital.
7. As you are now the supreme leader, you can do anything you want in your command. This game will reflect how you
9. REMEMBER! ALL YOUR CHOICES RESULTS DIFFERENT CONSEQUENCES
10. ##########################################
11. This is used for prototype purpose ONLY!
12. Real game will be applied to a game engine.
14. Dev Notes:
15. - Balanced randomize algorithm INTERGRATED
16. - Question banks algorithm NOT INTERGRATED
17. - Option buttons NOT WORKING
18. - District buttons are WORKING
19. """
21. **class** GameGUI:
22. **def** \_\_init\_\_(self, root):
23. """ Initialise tkinter root (for GUI only! variable declaration is seperated!) """
24. self.game\_init()
25. self.root = root
26. self.number = 0
27. self.numString = ''
29. # INITIALISE MAIN FRAME
30. self.frame\_top\_bar = Frame(self.root)
31. self.frame\_down\_info = Frame(self.root)
32. self.frame\_down\_map = Frame(self.root)
34. self.frame\_top\_bar.pack(fill=BOTH, expand=False)
35. self.frame\_down\_info.pack(side=LEFT, expand=True)
36. self.frame\_down\_map.pack(side=LEFT, fill=BOTH, expand=True)
38. # INITIALISE TOP BAR
39. self.label\_news = Label(self.frame\_top\_bar, text="NEWS: You are the son of a great dictator!")
40. self.label\_news.pack(fill=BOTH, expand=True)
42. # INITIALISE LEFT OPTIONS
43. self.email\_message = Label(self.frame\_down\_info, justify=LEFT, text="FROM: -\nTIME: -\nMessage: Not available")
44. self.frame\_option = Frame(self.frame\_down\_info)
45. self.frame\_option\_exp = Frame(self.frame\_down\_info)
46. self.label\_option\_1 = Label(self.frame\_option\_exp, text="This is the option 1 Explanation, \nplease check commander")
47. self.label\_option\_2 = Label(self.frame\_option\_exp, text="This is the option 2 Explanation, \nplease check commander")
48. self.button\_option\_1 = Button(self.frame\_option, text="Option 1")
49. self.button\_option\_2 = Button(self.frame\_option, text="Option 2")
50. self.district\_info = Label(self.frame\_down\_info, justify=LEFT,
51. text="District Unknown\n\nHappiness   : -\nPopulation  : - / -\nLow class   : -\n"
52. "Well-being  : -\nIncome/turn : FD. -\nFood Limit  : - / -")
54. self.email\_message.pack(side=TOP, fill=BOTH, expand=True)
55. self.frame\_option\_exp.pack(side=TOP, fill=BOTH, expand=True)
56. self.frame\_option.pack(side=TOP, fill=BOTH, expand=True)
57. self.label\_option\_1.pack(side=LEFT, fill=BOTH, expand=True)
58. self.label\_option\_2.pack(side=LEFT, fill=BOTH, expand=True)
59. self.button\_option\_1.pack(side=LEFT, fill=BOTH, expand=True)
60. self.button\_option\_2.pack(side=LEFT, fill=BOTH, expand=True)
61. self.district\_info.pack(side=TOP, fill=BOTH, expand=True)
63. # INITIALISE DISTRICT GUI STRUCTURE
64. self.frame1 = Frame(self.frame\_down\_map)
65. self.frame1.pack(fill=BOTH, expand=True)
66. self.frame2 = Frame(self.frame\_down\_map)
67. self.frame2.pack(fill=BOTH, expand=True)
68. self.frame3 = Frame(self.frame\_down\_map)
69. self.frame3.pack(fill=BOTH, expand=True)
70. self.frame4 = Frame(self.frame\_down\_map)
71. self.frame4.pack(fill=BOTH, expand=True)
72. self.frame5 = Frame(self.frame\_down\_map)
73. self.frame5.pack(fill=BOTH, expand=True)
74. self.frame6 = Frame(self.frame\_down\_map)
75. self.frame6.pack(fill=BOTH, expand=True)
77. # DECLARE DISTRICT BUTTONS
78. self.button\_d1 = Button(self.frame1, text='District 1',
79. command=**lambda**: self.district\_status\_update("District 1", self.district\_1.show\_string()))
80. self.button\_d2 = Button(self.frame1, text='District 2',
81. command=**lambda**: self.district\_status\_update("District 2", self.district\_2.show\_string()))
82. self.button\_d3 = Button(self.frame1, text='District 3',
83. command=**lambda**: self.district\_status\_update("District 3", self.district\_3.show\_string()))
84. self.button\_d4 = Button(self.frame2, text='District 4',
85. command=**lambda**: self.district\_status\_update("District 4", self.district\_4.show\_string()))
86. self.button\_d5 = Button(self.frame2, text='District 5',
87. command=**lambda**: self.district\_status\_update("District 5", self.district\_5.show\_string()))
88. self.button\_d6 = Button(self.frame2, text='District 6',
89. command=**lambda**: self.district\_status\_update("District 6", self.district\_6.show\_string()))
90. self.button\_d7 = Button(self.frame3, text='District 7',
91. command=**lambda**: self.district\_status\_update("District 7", self.district\_7.show\_string()))
92. self.button\_d8 = Button(self.frame3, text='District 8',
93. command=**lambda**: self.district\_status\_update("District 8", self.district\_8.show\_string()))
94. self.button\_d9 = Button(self.frame3, text='District 9',
95. command=**lambda**: self.district\_status\_update("District 9", self.district\_9.show\_string()))
96. self.button\_m1 = Button(self.frame4, text='Military 1',
97. command=**lambda**: self.district\_status\_update("Military 1", self.military\_1.show\_string()))
98. self.button\_m2 = Button(self.frame4, text='Military 2',
99. command=**lambda**: self.district\_status\_update("Military 2", self.military\_2.show\_string()))
100. self.button\_r1 = Button(self.frame4, text='Research',
101. command=**lambda**: self.district\_status\_update("Research", self.research.show\_string()))
102. self.button\_e1 = Button(self.frame5, text='Elite 1',
103. command=**lambda**: self.district\_status\_update("Elite 1", self.elite\_1.show\_string()))
104. self.button\_e2 = Button(self.frame5, text='Elite 2',
105. command=**lambda**: self.district\_status\_update("Elite 2", self.elite\_2.show\_string()))
106. self.button\_e3 = Button(self.frame5, text='Elite 3',
107. command=**lambda**: self.district\_status\_update("Elite 3", self.elite\_3.show\_string()))
108. self.button\_capital = Button(self.frame6, text='Capital',
109. command=**lambda**: self.district\_status\_update("Capital", self.capital.show\_string()))
111. # PACKING DISTRICTS
112. self.button\_d1.pack(side=LEFT, fill=BOTH, expand=True)
113. self.button\_d2.pack(side=LEFT, fill=BOTH, expand=True)
114. self.button\_d3.pack(side=LEFT, fill=BOTH, expand=True)
115. self.button\_d4.pack(side=LEFT, fill=BOTH, expand=True)
116. self.button\_d5.pack(side=LEFT, fill=BOTH, expand=True)
117. self.button\_d6.pack(side=LEFT, fill=BOTH, expand=True)
118. self.button\_d7.pack(side=LEFT, fill=BOTH, expand=True)
119. self.button\_d8.pack(side=LEFT, fill=BOTH, expand=True)
120. self.button\_d9.pack(side=LEFT, fill=BOTH, expand=True)
121. self.button\_m1.pack(side=LEFT, fill=BOTH, expand=True)
122. self.button\_m2.pack(side=LEFT, fill=BOTH, expand=True)
123. self.button\_r1.pack(side=LEFT, fill=BOTH, expand=True)
124. self.button\_e1.pack(side=LEFT, fill=BOTH, expand=True)
125. self.button\_e2.pack(side=LEFT, fill=BOTH, expand=True)
126. self.button\_e3.pack(side=LEFT, fill=BOTH, expand=True)
127. self.button\_capital.pack(fill=BOTH, expand=True)
129. **def** district\_status\_update(self, district\_name, district\_input):
130. """ Change string of District value when District pressed """
131. **try**:
132. district\_output = "%s\n\n%s" % (district\_name, district\_input)
133. self.district\_info.config(text=district\_output)
134. **except** Exception as e:
135. **print**("ERROR: " + str(e))
137. **def** combine\_funcs(\*funcs):
138. """ Tailored function combination method for tkinter future development """
139. **def** combined\_func(\*args, \*\*kwargs):
140. **for** f **in** funcs:
141. f(\*args, \*\*kwargs)
143. **return** combined\_func
145. **def** game\_init(self):
146. """ Initialise the game at start
147. note: this is a bad practice in programming but because of multiple redundant codes
148. I seperate this with the \_\_init\_\_
149. """
151. # Raw values
152. early\_population = randint(900000, 1000000)
153. early\_finance = randint(20000, 30000)
155. """ dividing each Districts with portion of values """
156. early\_pop\_normal = early\_population \* 0.7
157. early\_pop\_elite = early\_population \* 0.3
158. early\_fin\_normal = early\_finance \* 0.5
159. early\_fin\_elite = early\_finance \* 0.5
161. # district 1 - 9 variables
162. normal\_div = [0.1, 0.02, 0.18, 0.09, 0.11, 0.07, 0.12, 0.05, 0.15]
163. pop\_res, pop\_res\_cap, fin\_res, food\_res = [], [], [], []
165. # military, research and elite variables
166. el\_pop = [0.26, 0.24, 0.11, 0.13, 0.11, 0.12, 0.03]
167. el\_fin = [0.1, 0.12, 0.2, 0.08, 0.09, 0.11, 0.3]
168. el\_pop\_res, el\_pop\_res\_cap, el\_fin\_res, el\_food\_res = [], [], [], []
169. elite\_count = 1
171. """ append lists for easy assigning """
172. **for** percentage **in** normal\_div:
173. pop\_res.append(round(early\_pop\_normal \* percentage))
174. pop\_res\_cap.append(round(early\_pop\_normal \* percentage \* uniform(1, 1.2)))
175. food\_res.append(round(early\_pop\_normal \* percentage \* uniform(1, 1.3)))
176. fin\_res.append(round(early\_fin\_normal \* percentage, 2))
178. **for** percentage **in** el\_pop:
179. el\_pop\_res.append(round(early\_pop\_elite \* percentage))
180. **if** elite\_count <= 2:
181. el\_pop\_res\_cap.append(round(early\_pop\_elite \* percentage \* uniform(1.5, 2)))
182. el\_food\_res.append(round(early\_pop\_elite \* percentage \* uniform(1, 1.7)))
183. **elif** elite\_count == 3:
184. el\_pop\_res\_cap.append(round(early\_pop\_elite \* percentage \* uniform(1.3, 1.7)))
185. el\_food\_res.append(round(early\_pop\_elite \* percentage \* uniform(1, 1.7)))
186. **elif** 4 <= elite\_count <= 6:
187. el\_pop\_res\_cap.append(round(early\_pop\_elite \* percentage \* uniform(2, 2.8)))
188. el\_food\_res.append(round(early\_pop\_elite \* percentage \* uniform(2, 3)))
189. **elif** elite\_count == 7:
190. el\_pop\_res\_cap.append(round(early\_pop\_elite \* percentage \* uniform(2.5, 4)))
191. el\_food\_res.append(round(early\_pop\_elite \* percentage \* uniform(5, 8)))
192. elite\_count += 1
194. **for** percentage **in** el\_fin:
195. el\_fin\_res.append(round(early\_fin\_elite \* percentage, 2))
197. """ create an object for each Districts with determined variables """
198. # population, population\_cap, inequalities, happiness, health, finance, food\_limit
199. self.district\_1 = District(pop\_res[0], pop\_res\_cap[0], randint(55, 99), randint(60, 85),
200. randint(65, 85), fin\_res[0], food\_res[0])
201. self.district\_2 = District(pop\_res[1], pop\_res\_cap[1], randint(55, 99), randint(60, 85),
202. randint(65, 85), fin\_res[1], food\_res[1])
203. self.district\_3 = District(pop\_res[2], pop\_res\_cap[2], randint(55, 99), randint(60, 85),
204. randint(65, 85), fin\_res[2], food\_res[2])
205. self.district\_4 = District(pop\_res[3], pop\_res\_cap[3], randint(55, 99), randint(60, 85),
206. randint(65, 85), fin\_res[3], food\_res[3])
207. self.district\_5 = District(pop\_res[4], pop\_res\_cap[4], randint(55, 99), randint(60, 85),
208. randint(65, 85), fin\_res[4], food\_res[4])
209. self.district\_6 = District(pop\_res[5], pop\_res\_cap[5], randint(55, 99), randint(60, 85),
210. randint(65, 85), fin\_res[5], food\_res[5])
211. self.district\_7 = District(pop\_res[6], pop\_res\_cap[6], randint(55, 99), randint(60, 85),
212. randint(65, 85), fin\_res[6], food\_res[6])
213. self.district\_8 = District(pop\_res[7], pop\_res\_cap[7], randint(55, 99), randint(60, 85),
214. randint(65, 85), fin\_res[7], food\_res[7])
215. self.district\_9 = District(pop\_res[8], pop\_res\_cap[8], randint(55, 99), randint(60, 85),
216. randint(65, 85), fin\_res[8], food\_res[8])
218. self.military\_1 = District(el\_pop\_res[0], el\_pop\_res\_cap[0], randint(3, 8), randint(80, 95),
219. randint(90, 98), el\_fin\_res[0], el\_food\_res[0])
220. self.military\_2 = District(el\_pop\_res[1], el\_pop\_res\_cap[1], randint(3, 8), randint(80, 95),
221. randint(90, 98), el\_fin\_res[1], el\_food\_res[1])
222. self.research = District(el\_pop\_res[2], el\_pop\_res\_cap[2], randint(0, 3), randint(80, 95),
223. randint(90, 98), el\_fin\_res[2], el\_food\_res[2])
224. self.elite\_1 = District(el\_pop\_res[3], el\_pop\_res\_cap[3], randint(1, 5), randint(90, 95),
225. randint(90, 98), el\_fin\_res[3], el\_food\_res[3])
226. self.elite\_2 = District(el\_pop\_res[4], el\_pop\_res\_cap[4], randint(1, 5), randint(90, 95),
227. randint(90, 98), el\_fin\_res[4], el\_food\_res[4])
228. self.elite\_3 = District(el\_pop\_res[5], el\_pop\_res\_cap[5], randint(1, 5), randint(90, 95),
229. randint(90, 98), el\_fin\_res[5], el\_food\_res[5])
230. self.capital = District(el\_pop\_res[6], el\_pop\_res\_cap[6], randint(0, 1), 100,
231. randint(97, 100), el\_fin\_res[6], el\_food\_res[6])

234. **class** District:
235. **def** \_\_init\_\_(self, population, population\_cap, inequalities, happiness, health, finance, food\_limit):
236. """ initialise Tkinter object with title and screen size """
237. self.population = population
238. self.population\_cap = population\_cap
239. self.inequalities = inequalities
240. self.happiness = happiness
241. self.health = health
242. self.finance = finance
243. self.food\_limit = food\_limit
244. self.property\_values = [self.population, self.population\_cap, self.inequalities, self.happiness, self.health,
245. self.finance, self.food\_limit]
247. """ methods begin with 'edit' are used to change District's var value when Option(not exist yet) is choosen """
248. **def** edit\_population(self, state, value):
249. **if** state:
250. self.population += value
251. **elif** **not** state:
252. self.population -= value
254. **def** edit\_inequalities(self, state, value):
255. **if** state:
256. self.inequalities += value
257. **elif** **not** state:
258. self.inequalities -= value
260. **def** edit\_happiness(self, state, value):
261. **if** state:
262. self.happiness += value
263. **elif** **not** state:
264. self.happiness -= value
266. **def** edit\_health(self, state, value):
267. **if** state:
268. self.health += value
269. **elif** **not** state:
270. self.health -= value
272. **def** edit\_finance(self, state, value):
273. **if** state:
274. self.happiness += value
275. **elif** **not** state:
276. self.happiness -= value
278. **def** edit\_food\_limit(self, state, value):
279. **if** state:
280. self.food\_limit += value
281. **elif** **not** state:
282. self.food\_limit -= value
284. **def** show\_stats(self):
285. """ return a list of District variables """
286. **return** self.property\_values
288. **def** show\_string(self):
289. """ set string to be displayed on UI """
290. property\_string = "Happiness     : %s%%\n" \
291. "Population    : %s / %s\n" \
292. "Low class       : %s%%\n" \
293. "Well-being    : %s%%\n" \
294. "Income/turn : FD. %s\n" \
295. "Food Limit     : %s" % (self.happiness, self.population, self.population\_cap,
296. self.inequalities, self.health, self.finance, self.food\_limit)
297. **return** property\_string

300. **def** main():
301. """ initialise Tkinter object with title and screen size """
302. root = Tk()
303. md\_ver = str(0.2)
304. root.wm\_title("Moral Dictator %s" % md\_ver)
305. root.geometry('%dx%d' % (700, 400))
306. gui = GameGUI(root)
307. root.mainloop()
309. **if** \_\_name\_\_ == '\_\_main\_\_':
310. sys.exit(main())

**Prototype:**



As we switch between different districts we can see that each one of them has different statistics. This means that some answers will have a higher impact in some districts than in others.