SOFTWARE ENGINEERING USE CASES

ON THE PROJECT OF

"Clinical Diagnosis iOS App - The Phenomizer"

Submitted to

Dr. Marcel Schulz
Multimodal Computing and Interaction
Cluster of Excellence
Saarland University
Campus E1 4 Room 515
66123, Saarbrucken
Germany

BY

Ali Shah - 2552956
Christopher Chen - 2561030
Margarita Salyaeva - 2556501
Muhammad Shoaib Malik - 2552987
Nadia Ashraf - 2557949
Robert Wicks - 2560407
Waqas Durrani - 2558024

UNDER THE GUIDANCE OF

Emamurho Juliet Ugherughe

Project Description

The project is about building and implementing an iOS App "The phenomizer" to facilitate the disease prediction from a set of phenotypes. The App is intended for medical doctors and performs computational clinical diagnosis to return the results to the doctor. Through the App, the doctor can search or select one or more phenotype(s) from a list and these phenotype(s) are queried against a database of known diseases that are associated with the selected phenotype(s). The diseases are then ranked upon the likelihood of them being caused by the queried phenotype(s). Results are then displayed to the doctor.

Contents

1 Getting diagnosis by selecting phenotype(s) from features tab

Primary actor: Doctor.

Goal in context: To fetch the disease(s) that could result from the selected phenotype(s) given as the search query. The database consists of known diseases that are associated with different phenotype(s).

Preconditions:

- Internet connection is available.
- Either minimum or full update of the database has been performed.
- Doctor is currently at the ***Welcome*** screen.

Trigger: Doctor wants to get the diagnosis by selecting phenotype(s) going through the complete list of all phenotypes.

Scenario:

1. Doctor observes the ***Welcome*** screen.
2. Doctor presses the ***New Query*** button on the welcome screen.
3. Doctor is directed to a ***Features*** tab listing all available phenotypes present in the database.
4. Doctor selects phenotype(s).
5. Doctor presses the ***Show Selection*** tab.
6. Doctor observes the phenotype(s) he or she selected.
7. Doctor presses the ***Perform Diagnosis*** tab.
8. A list of diseases ranked upon the likelihood of them being caused by the selected phenotype(s) is then displayed to the doctor.

Exceptions:

- No internet connection is available. An error alert is displayed on the screen.
- Database is never updated. The doctor is shown empty phenotype list when he or she is directed to ***Features*** tab from the welcome screen.

Priority: High. To be implemented as a basic feature of the application.

When available: Second iteration.

Frequency of use: Pretty frequent as this is the main feature of the application.

Channel to actor: Via ***Welcome*** screen interface.

Secondary actors: The phenomizer server/database containing known diseases that are associated with different phenotype(s).

Channels to secondary actors: Internet.

Open issues:

- Is it necessary for the database present on the iOS to be up-to-date with the database present on the server?
- Can diagnosis be performed without using the internet?

2 Getting diagnosis by searching for phenotype's name or HPO id

Primary actor: Doctor.

Goal in context: To fetch the disease(s) that could result from the selected phenotype(s) given as the search query. The database consists of known diseases that are associated with different phenotype(s).

Preconditions:

- Internet connection is available.
- Either minimum or full update of the database has been performed.
- Doctor is currently at the ***Welcome*** screen.

Trigger: Doctor wants to get the diagnosis by searching for name or HPO id of the desired phenotype instead of going through the complete list of phenotypes.

Scenario:

1. Doctor observes the ***Welcome*** screen.

2. Doctor presses the ***New Query*** button on the welcome screen.
3. Doctor is directed to a ***Features*** tab listing all available phenotypes present in the database.
4. Doctor types the name or HPO id of his or her desired phenotype in the search field present on the Features tab.
5. Doctor submits the query.
6. The phenotype matching the doctor's query is displayed on the screen.
7. Doctor selects the phenotype.
8. Doctor repeats step 4-7 and performs the search again if he or she wants to select more phenotype(s).
9. Doctor presses the ***Show Selection*** tab.
10. Doctor observes the phenotype(s) he or she selected.
11. Doctor presses the ***Perform Diagnosis*** tab.
12. A list of diseases ranked upon the likelihood of them being caused by the selected phenotype(s) is then displayed to the doctor.

Exceptions:

- No internet connection is available. An error alert is displayed on the screen.
- The queried phenotype is not present on the doctor's iOS system. The doctor is shown empty phenotype list when he or she is finished entering his or her query.
- Database is never updated. The doctor is shown empty phenotype list when he or she is directed to ***Features*** tab from the welcome screen.

Priority: Moderate. To be implemented after the basic features of the application.

When available: Third iteration.

Frequency of use: Moderate.

Channel to actor: Via ***Feature*** screen interface.

Secondary actors: The phenomizer server/database containing known diseases that are associated with different phenotype(s).

Channels to secondary actors: Internet.

Open issues:

- Is it necessary for the database present on the iOS to be up-to-date with the database present on the server?
- Can diagnosis be performed without using the internet?

3 Getting diagnosis by selecting phenotype(s) from diseases tab

Primary actor: Doctor.

Goal in context: To fetch the disease(s) that could result from the selected phenotype(s) given as the search query. The database consists of known diseases that are associated with different phenotype(s).

Preconditions:

- Internet connection is available.
- Full update of the database has been performed.
- Doctor is currently at the ***Welcome*** screen.

Trigger: Doctor wants to select phenotype(s) from diseases that are caused as a result of those phenotype(s).

Scenario:

1. Doctor observes the ***Welcome*** screen.
2. Doctor presses the ***New Query*** button on the welcome screen.
3. Doctor is directed to a ***Features*** tab listing all available phenotypes present in the database.
4. Doctor switches to ***Diseases*** tab from the ***Features*** tab.
5. Doctor observes the ***Diseases*** screen.
6. Doctor presses on the desired disease among those listed on the diseases screen.
7. Doctor is directed to a new screen where the phenotype(s) that cause that particular disease are displayed.
8. Doctor selects phenotype(s).
9. Doctor repeats step 6-8 and presses on more diseases if he or she wants to select more phenotype(s).
10. Doctor presses the ***Show Selection*** tab.

11. Doctor observes the list of all the phenotype(s) he or she selected.
12. Doctor presses the ***Perform Diagnosis*** tab.
13. A list of diseases ranked upon the likelihood of them being caused by the selected phenotype(s) is then displayed to the doctor.

Exceptions:

- No internet connection is available. An error alert is displayed on the screen.
- Complete update of database has never been performed. The doctor is shown empty diseases list when he or she is directed to ***Diseases*** screen from the ***Features*** screen.

Priority: High. To be implemented as the basic feature of the application.

When available: Second iteration.

Frequency of use: Moderate.

Channel to actor: Via ***Diseases*** screen interface.

Secondary actors: The phenomizer server/database containing known diseases that are associated with different phenotype(s).

Channels to secondary actors: Internet.

Open issues:

- Is it necessary for the database present on the iOS to be up-to-date with the database present on the server?
- Can diagnosis be performed without using the internet?

4 Getting diagnosis by searching diseases from diseases tab

Primary actor: Doctor.

Goal in context: To fetch the disease(s) that could result from the selected phenotype(s) given as the search query. The database consists of known diseases that are associated with different phenotype(s).

Preconditions:

- Internet connection is available.
- Full update of the database has been performed.
- Doctor is currently at the ***Diseases*** screen.

Trigger: Doctor wants to get the diagnosis by searching for desired disease and then selecting desired phenotype instead of going through the complete list of diseases.

Scenario:

1. Doctor observers the ***Diseases*** screen.
2. Doctor types the name of his or her desired disease in the search field present on the Diseases tab.
3. Doctor submits the query.
4. The diseases matching the doctor's query is displayed on the screen.
5. Doctor presses on the desired disease among those listed on the diseases screen.
6. Doctor is directed to a new screen where the phenotype(s) that cause that particular disease are displayed.
7. Doctor selects phenotype(s).
8. Doctor repeats step 2-7 and presses on more diseases if he or she wants to select more phenotype(s).
9. Doctor presses the ***Show Selection*** tab.
10. Doctor observes the list of all the phenotype(s) he or she selected.
11. Doctor presses the ***Perform Diagnosis*** tab.
12. A list of diseases ranked upon the likelihood of them being caused by the selected phenotype(s) is then displayed to the doctor.

Exceptions:

- No internet connection is available. An error alert is displayed on the screen.
- Complete update of database has never been performed. The doctor is shown empty diseases list when he or she is directed to ***Diseases*** screen from any other screen.

Priority: Moderate. To be implemented after the basic features of the application.

When available: Third iteration.

Frequency of use: Moderate.

Channel to actor: Via ***Diseases*** screen interface.

Secondary actors: The phenomizer server/database containing known diseases that are associated with different phenotype(s).

Channels to secondary actors: Internet.

Open issues:

- Is it necessary for the database present on the iOS to be up-to-date with the database present on the server?
- Can diagnosis be performed without using the internet?

5 Adding more phenotypes(s) to list of selected phenotype(s)

Primary actor: Doctor.

Goal in context: To add more phenotype(s) to an existing list of selected phenotypes(s) to fetch the disease(s) that could result from these selected phenotype(s).

Preconditions:

- Internet connection is available.
- Either minimum or full update of the database has been performed.
- Doctor is currently at the ***Show Selection*** screen.

Trigger: Doctor selects his or her desired phenotype(s) from ***Features*** screen and goes to ***Show Selection*** screen but now wants to add some more phenotype(s) to this list of selected phenotype(s).

Scenario:

1. Doctor observes the selected phenotype(s) listed on the ***Show Selection*** screen.
2. Doctor presses the ***Add Features*** tab on the ***Show Selection*** screen.
3. Doctor is directed to a ***Features*** or ***Diseases*** tab depending on how he or she reached the ***Show Selection*** screen.

4. Doctor selects more phenotype(s) if he or she wishes.

Exceptions:

- Complete update of database is never performed. The doctor is shown empty list of diseases when he or she is directed to ***Diseases*** screen from the ***Show Selection*** window.
- Minimum update of database is never performed. The doctor is shown empty list of phenotype(s) when he or she is directed to ***Features*** screen from the ***Show Selection*** window.

Priority: High. To be implemented as the basic feature of the application.

When available: Second iteration.

Frequency of use: High.

Channel to actor: Via ***Show Selection*** screen interface.

Secondary actors: The phenomizer server/database containing the list of features and ontologies.

Channels to secondary actors: Internet.

Open issues:

- Is it necessary to direct the doctor to ***Features*** or ***Diseases*** screen depending on how he or she reached the ***Show Selection*** screen?

Note: Client made a comment to use the ***Back*** button (instead of ***Add Features*** button) to go to previous screen from ***Show Selection*** screen to use this functionality. Hence the ***Add Features*** button is no more available.

6 Getting full update of database

Primary actor: Doctor.

Goal in context: To get the latest version of the complete database on the iOS system.

Preconditions:

- Internet connection is available.

- Enough disk space is available on doctor's iOS system.
- Doctor is currently at the ***Welcome*** screen.

Trigger: Doctor wants to have the most up-to-date list of phenotypes, ontologies, diseases and annotations on his or her iOS system.

Scenario:

1. Doctor observes the ***Welcome*** screen.
2. Doctor presses the ***Update Database*** button on the welcome screen.
3. Doctor is directed to a new screen with options of ***Minimum Update*** and ***Full Update***.
4. Doctor clicks on the ***Full Update*** button.
5. The most recent list of phenotypes, ontologies, diseases and annotations present at the server/database is downloaded onto the doctor's iOS system.

Exceptions:

- No internet connection is available. An error alert is displayed on the screen.
- Not enough disk space is available on doctor's iOS system. An error alert is displayed on the screen.
- Internet connection is interrupted in the middle of download. An error alert is displayed on the screen.

Priority: High. To be implemented as a basic feature of the application.

When available: Second iteration.

Frequency of use: Moderate.

Channel to actor: Via ***Welcome*** screen interface.

Secondary actors: The phenomizer server/database containing the list of features and ontologies.

Channels to secondary actors: Internet.

Open issues:

- Is it necessary to display error alert on the screen when no internet connection is available?
- Is it necessary to display error alert on the screen when internet connection is interrupted in the middle of download?

7 Getting minimum update of database

Primary actor: Doctor.

Goal in context: To get the latest version of the minimal database on the iOS system.

Preconditions:

- Internet connection is available.
- Enough disk space is available on doctor's iOS system.
- Doctor is currently at the ***Welcome*** screen.

Trigger: Doctor wants to have the most up-to-date list of phenotype(s) and ontologies on his or her iOS system.

Scenario:

1. Doctor observes the ***Welcome*** screen.
2. Doctor presses the ***Update Database*** button on the welcome screen.
3. Doctor is directed to a new screen with options of ***Minimum Update*** and ***Full Update***.
4. Doctor clicks on the ***Minimum Update*** button.
5. The most recent list of phenotype(s) and ontologies present at the server/database is downloaded onto the doctor's iOS system.

Exceptions:

- No internet connection is available. An error alert is displayed on the screen.
- Not enough disk space is available on doctor's iOS system. An error alert is displayed on the screen.
- Internet connection is interrupted in the middle of download. An error alert is displayed on the screen.

Priority: Moderate. To be implemented after the basic features of the application.

When available: Third iteration.

Frequency of use: Moderate.

Channel to actor: Via ***Welcome*** screen interface.

Secondary actors: The phenomizer server/database containing the list of features and ontologies.

Channels to secondary actors: Internet.

Open issues:

- Is it necessary to display error alert on the screen when no internet connection is available?
- Is it necessary to display error alert on the screen when internet connection is interrupted in the middle of download?

8 Help

Primary actor: Doctor.

Goal in context: To understand how to get diagnosis.

Preconditions:

- Internet connection is available.
- Doctor is currently at the ***Welcome*** screen.

Trigger: Doctor has no or limited prior experience of using the App wants to get information on how to go about getting the diagnosis.

Scenario:

1. Doctor observes the ***Welcome*** screen.
2. Doctor presses the ***Help*** button on the welcome screen.
3. Doctor is directed to a Help page listing all the necessary information from start to finish on how to get diagnosis against a list of selected phenotype(s).

Exceptions:

- No internet connection is available. An error alert is displayed on the screen.

Priority: Moderate. To be implemented after the basic features of the application.

When available: Second iteration.

Frequency of use: Seldom.

Channel to actor: Via ***Welcome*** screen interface.

Secondary actors: The phenomizer server/database containing the list of features and ontologies.

Channels to secondary actors: Internet.

Open issues:

- Is it necessary to be connected to the internet to view the contents of the ***Help*** Screen.

9 Save Query

Primary actor: Doctor.

Goal in context: To save a list of phenotype(s) to be used later for diagnosis.

Preconditions:

- Either minimum or full update of the database has been performed.
- Doctor is currently at the ***Show Selection*** screen.

Trigger: Doctor wants to save the list of selected phenotype(s) if he or she wishes to re-use the phenotype(s) later for diagnosis.

Scenario:

1. Doctor observes the list of selected phenotype(s) at the ***Show Selection*** screen.
2. Doctor presses the ***Save*** tab.
3. A dialog box appears on the screen.
4. Doctor enters a name for the query.
5. Doctor presses the ***Save*** option on the dialog box.
6. The query which is a list of selected phenotype(s) is saved on doctor's iOS system.

Exceptions:

- Database is never updated. An empty phenotype list is saved on doctor's iOS system.

Priority: Moderate. To be implemented after the basic features of the application.

When available: Third iteration.

Frequency of use: Moderate.

Channel to actor: Via ***Show Selection*** screen interface.

Secondary actors: The phenomizer server/database containing known diseases that are associated with different phenotype(s).

Channels to secondary actors: Internet.

Open issues:

- Is it necessary for the empty phenotype list to be saved on doctor's iOS system?
- Is it necessary for a query with no name to be saved on doctor's iOS system?

10 Load Query

Primary actor: Doctor.

Goal in context: To load a list of phenotype(s) and use it for diagnosis.

Preconditions:

- Either minimum or full update of the database has been performed.
- Doctor is currently at the ***Welcome*** screen.

Trigger: Doctor wants to load the list of selected phenotype(s) and wishes to use this list for diagnosis.

Scenario:

1. Doctor observes the ***Welcome*** screen.
2. Doctor presses the ***Load Query*** button on the welcome screen.
3. Doctor is directed to a Load Query page listing all the saved queries.

4. Doctor presses on the desired query.
5. Doctor is directed to ***Show Selection*** screen displaying the phenotype(s) present in the saved query.

Exceptions:

- Database is never updated. Doctor is unable to select a query from ***Load Query*** screen which contains any phenotype(s).

Priority: Moderate. To be implemented after the basic features of the application.

When available: Third iteration.

Frequency of use: Moderate.

Channel to actor: Via ***Welcome*** screen interface.

Secondary actors: The phenomizer server/database containing known diseases that are associated with different phenotype(s).

Channels to secondary actors: Internet.

Open issues:

- Is it necessary for the empty phenotype list to be saved on doctor's iOS system?

11 Edit Query

Primary actor: Doctor.

Goal in context: To edit an existing saved query.

Preconditions:

- Either minimum or full update of the database has been performed.
- Doctor is currently at the ***Welcome*** screen.

Trigger: Doctor wants to edit the list of selected phenotype(s), saved in a query, and wishes to use this edited list for diagnosis.

Scenario:

1. Doctor observes the ***Welcome*** screen.
2. Doctor presses the ***Load Query*** button on the welcome screen.
3. Doctor is directed to a Load Query page listing all the saved queries.
4. Doctor presses on the desired query.
5. Doctor is directed to ***Show Selection*** screen displaying the phenotype(s) present in the saved query.
6. Doctor performs changes to the list of selected Phenotype(s).
7. Doctor presses the ***Save*** tab.
8. A dialog box appears on the screen.
9. Doctor enters the name for the changed query.
10. Doctor presses Overwrite.
11. The query, which is a list of selected phenotype(s),is saved on doctor's iOS system under the given name.

Exceptions:

- Database is never updated. Doctor is unable to select a query from ***Load Query*** screen which contains any phenotype(s).

Priority: Moderate. To be implemented after the basic features of the application.

When available: Third iteration.

Frequency of use: Moderate.

Channel to actor: Via ***Welcome*** screen interface.

Secondary actors: The phenomizer server/database containing known diseases that are associated with different phenotype(s).

Channels to secondary actors: Internet.

Open issues:

- Is it necessary for the empty phenotype list to be saved on doctor's iOS system?
- Is it necessary for a query with no name to be saved on doctor's iOS system?