Module Interface Specification for Software Engineering

Team 14, Reach
Aamina Hussain
David Morontini
Anika Peer
Deep Raj
Alan Scott

January 15, 2024

1 Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at [give url —SS]

[Also add any additional symbols, abbreviations or acronyms —SS]

Contents

3 Introduction

The following document details the Module Interface Specifications for REACH, a web application used to improve patients' access to clinical trials and practitioners' access to potential participants. More specifically, it will provide the list of modules that have been decomposed from the Module Guide, each with their interface specification, detailing important characteristics such as the module's methods and state variables.

Complementary documents, such as the System Requirement Specifications and Module Guide can be found at https://github.com/davimang/REACH.

4 Notation

The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1|c_2 \Rightarrow r_2|...|c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by Software Engineering.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	$\mathbb{Z}$	a number without a fractional component in (-∞, ∞)
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)

The specification of Software Engineering uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, Software Engineering uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	**Level 2**
Hardware-Hiding	
Behaviour-Hiding	Input Parameters Output Format Output Verification Temperature ODEs Energy Equations Control Module Specification Parameters Module
Software Decision	Sequence Data Structure ODE Solver Plotting

Table 1: Module Hierarchy

6 MIS of the User data module

6.1 Module

User

6.2 Uses

PatientInfo, Trial

6.3 Syntax

6.3.1 Exported Constants

None

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
getName	-	seq of char	-
setName	seq of char	-	EmptyName
getEmail	-	seq of char	-
setEmail	seq of char	-	InvalidEmail
getInfoProfiles	-	list of InfoProfile	-
getInfoProfile	integer	InfoProfile	InvalidInfoProfileId
addInfoProfile	InfoProfile	-	-
removeInfoProfile	integer	-	-
addTrial	Trial	-	-
removeTrial	integer	-	-
getTrials	-	list of Trial	-
getTrial	integer	Trial	InvalidTrialId

6.4 Semantics

6.4.1 State Variables

name: seq of char
email: seq of char
infoProfiles: list of PatientInfo
trials: list of Trial

6.4.2 Environment Variables

None

6.4.3 Assumptions

- Each InfoProfile has a unique id.
- Each Trial has a unique id.

6.4.4 Access Routine Semantics

getName():

- transition: N/A
- output: out := self.name
- exception: N/A

setName(newName: seq of char):

- transition: self.name := newName
- output: N/A
- exception: $exc := length(newName) == 0 \Rightarrow EmptyName$

getEmail():

- transition: N/A
- output: out := self.email
- exception: N/A

setEmail(newEmail: seq of char):

- transition: self.email := newEmail
- output: N/A
- exception: $exc := isInvalidEmail(newEmail) \Rightarrow InvalidEmail$

getInfoProfiles():

- transition: N/A
- output: out := self.infoProfiles
- exception: N/A

getInfoProfile(id: integer):

- transition: N/A

- output: out := $\{\exists i \in self.infoProfiles | i.id = id\} \Rightarrow i$
- exception: exc := $\neg\{\exists i \in self.infoProfiles | i.id = id\} \Rightarrow InvalidInfoProfileId$

addInfoProfile(newInfoProfile: InfoProfile):

- transition: self.infoProfiles = self.infoProfiles + newInfoProfile (add the new infoprofile to the list of info profiles connected to the current user)
- output: N/A
- exception: N/A

removeInfoProfile(oldInfoProfile: InfoProfile):

- transition: self.infoProfiles = self.infoProfiles - oldInfoProfile (remove the info profile passed to the method from the list of info profiles connected to the current user)
- output: N/A
- exception: N/A

getTrials():

- transition: N/A
- output: out := self.trials
- exception: N/A

getTrial(id: integer):

- transition: N/A
- output: out := $\{\exists i \in self.trials | i.id = id\} \Rightarrow i$
- exception: exc := $\neg\{\exists i \in self.trials | i.id = id\} \Rightarrow InvalidTrialId$

addTrial(newTrial: Trial):

- transition: self.trials = self.trials + newTrial (same idea as add info profiles)
- output: N/A
- exception: N/A

removeTrial(oldTrial: Trial):

- transition: self.trials = self.trials - oldTrial (same idea as remove info profiles)
- output: N/A
- exception: N/A

6.4.5 Local Functions

saveUser(user: User):

- transition: Saves the user to the database.
- output: N/A
- exception: N/A

loadUser(user: User):

- transition: Loads the user from the database.
- output: N/A
- exception: N/A

7 MIS of Info Profile data module

7.1 Module

InfoProfile

7.2 Uses

None

7.3 Syntax

7.3.1 Exported Constants

None

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
getDOB	-	datetime	-
setDOB	datetime	-	-
getAddress	-	seq of char	-
setAddress	seq of char	-	-
getGender	-	seq of char	-
setGender	seq of char	-	-
getHealthDetails	-	map<seq of char: Any>	-
setHealthDetails	map<seq of char: Any>	-	-

7.4 Semantics

7.4.1 State Variables

dateOfBirth: datetime
address: seq of char
gender: seq of char
healthDetails: map<seq of char: Any>

7.4.2 Environment Variables

None

7.4.3 Assumptions

None

7.4.4 Access Routine Semantics

getDOB():

- transition: N/A
- output: out := self.dateOfBirth
- exception: N/A

setDOB(newDOB: datetime):

- transition: self.dateOfBirth = newDOB
- output: N/A
- exception: N/A

getAddress():

- transition: N/A
- output: out := self.address
- exception: N/A

setAddress(newAddress: seqOfChar):

- transition: self.address = newAddress
- output: N/A
- exception: N/A

getGender():

- transition: N/A
- output: out := self.gender
- exception: N/A

setGender(gender: seq of char):

- transition: self.gender = gender
- output: N/A
- exception: N/A

getHealthDetails():

- transition: N/A
- output: out := self.healthDetails
- exception: N/A

setHealthDetails(newHealthDetails: map<seq of char: Any>):

- transition: self.healthDetails = newHealthDetails
- output: N/A
- exception: N/A

7.4.5 Local Functions

saveInfoProfile(infoProfile: InfoProfile):

- transition: Saves the info profile to the database.
- output: N/A
- exception: N/A

loadInfoProfile(infoProfile: InfoProfile):

- transition: Loads the info profile from the database.
- output: N/A
- exception: N/A

8 MIS of Trial data module

8.1 Module

Trial

8.2 Uses

None

8.3 Syntax

8.3.1 Exported Constants

None

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
getTitle	-	seq of char	-
getDescription	-	seq of char	-
getUrl	-	seq of char	-

8.4 Semantics

8.4.1 State Variables

title: seq of char
description: seq of char
url: seq of char

8.4.2 Environment Variables

None

8.4.3 Assumptions

- Trial details (title, description, url) will not need to be changed. Therefore no need for setter methods to update these values for a certain trial.

8.4.4 Access Routine Semantics

getTitle():

- transition: N/A

- output: out := self.title
- exception: N/A

getDescription():

- transition: N/A
- output: out := self.description
- exception: N/A

getUrl():

- transition: N/A
- output: out := self.url
- exception: N/A

8.4.5 Local Functions

saveTrial(trial: Trial):

- transition: Saves the trial to the database.
- output: N/A
- exception: N/A

loadTrial(trial: Trial):

- transition: Loads the trial from the database.
- output: N/A
- exception: N/A

9 MIS of the Fetch Trials Modules

9.1 Module

TrialFetcher

9.2 Uses

Trial

9.3 Syntax

9.3.1 Exported Constants

None

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
getTrials	seq. of String, integer, String	DataFrame	MissingParameter, InvalidAge, InvalidAddress
getLocator	-	geocoder	-
setLocator	geocoder	-	-

9.4 Semantics

9.4.1 State Variables

locator: geocoder

9.4.2 Environment Variables

None

9.4.3 Assumptions

- Each Trial has a unique id.
- The trial API will always be accessible.

9.4.4 Access Routine Semantics

getTrials(conditions: sequence of String, age: int, address: String):

- transition: None
- output: out := DataFrame populated with trials from the ClinicalTrials.gov API
- exception: $exc := (age \notin (0, 120] \rightarrow InvalidAge) \lor (\neg checkAddress(address) \rightarrow InvalidAddress) \lor ((\exists x.x \in parameters : x = \varepsilon) \rightarrow MissingParameter)$

9.4.5 Local Functions

convertTrialsToDataFrame(rawData: csv):

- transition: None
- output: out := rawData formatted as a DataFrame
- exception: None

checkAddress(address: String):

- transition: None
- output: out:= True
- exception: $exc := (geopy.geolocator(address) = exception \rightarrow False)$

10 MIS of the Trial Filtering Module

10.1 Module

TrialFilterer

10.2 Uses

Trial

10.3 Syntax

10.3.1 Exported Constants

None

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
exportTrials	-	json	-
fetchTrials	seq. of String, integer, String	-	-
getLocator	-	geocoder	-
setLocator	geocoder	-	-

10.4 Semantics

10.4.1 State Variables

locator: geocoder
trials: DataFrame

10.4.2 Environment Variables

None

10.4.3 Assumptions

- Each Trial has a unique id.
- Exceptions are caught downstream by the TrialFetcher module

10.4.4 Access Routine Semantics

fetchTrials(conditions: sequence of String, age: int, address: String):

- transition: self.trials populated with trials via TrialFilterer module
- output: None
- exception: None

exportTrials():

- transition: None
- output: out:= self.trials as json
- exception: None

getLocator():

- transition: None

- output: out:= self.locator
- exception: None

setLocator(loc: geocoder):

- transition: self.locator = loc
- output: None
- exception: None

10.4.5 Local Functions

cleanAge(stringAge: String):

- transition: None
- output: out:= $(inMonths \rightarrow int(stringAge)/12) \rightarrow int(stringAge)$
- exception: $exc := stringAge \notin \mathbb{R} \rightarrow InvalidAge$

geodesicDistance(address: geocode, trialLocation: geocode):

- transition: None
- output: $out :=$ arccos(sin($address.latitude$)·sin($trialLocation.latitude$)+cos($address.latitude$)·cos($ltrialLocation.latitude$)·cos($trialLocation.longitude - address.longitude$))·6371000
- exception: None

calculateDistance():

- transition: $self.trials[distance] \mapsto geodesicDistance(address, self.trials[trialLocation])$
- output: None
- exception: None

convertToJSON(df: DataFrame):

- transition: None
- output: $df \rightarrow json(df)$
- exception: None

11 MIS of the Registration Module

11.1 Module

Registration

11.2 Uses

User

11.3 Syntax

11.3.1 Exported Constants

None

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
registerUser	String, String	Boolean	-

11.4 Semantics

11.4.1 State Variables

11.4.2 Environment Variables

11.4.3 Assumptions

None

11.4.4 Access Routine Semantics

registerUser(emailAddress: String, password: String):

- transition: None
- output: out := True if the Registration was successful, False otherwise
- exception: None

11.4.5 Local Functions

12 MIS of the Login Module

12.1 Module

Login

12.2 Uses

User

12.3 Syntax

12.3.1 Exported Constants

None

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
loginUser	String, String	Boolean	-

12.4 Semantics

12.4.1 State Variables

12.4.2 Environment Variables

12.4.3 Assumptions

None

12.4.4 Access Routine Semantics

loginUser(emailAddress: String, password: String):

- transition: None
- output: out := True if the login was successful, False otherwise
- exception: None

12.4.5 Local Functions

13 MIS of the Email Template Module

13.1 Module

EmailTemplate

13.2 Uses

User, Trial

13.3 Syntax

13.3.1 Exported Constants

None

13.3.2 Exported Access Programs

Name	In	Out	Exceptions
createEmail	User, Trial	String	-

13.4 Semantics

13.4.1 State Variables

13.4.2 Environment Variables

13.4.3 Assumptions

13.4.4 Access Routine Semantics

createEmail(user: User, trial: Trial):

- transition: None
- output: out := email template personalized using User and Trial data
- exception: None

13.4.5 Local Functions

14 MIS of the Email Notification Module

14.1 Module

NotificationModule, User, PatientInfo, Trial

14.2 Uses

14.3 Syntax

14.3.1 Exported Constants

None

14.3.2 Exported Access Programs

Name	In	Out	Exceptions
sendEmail	String, String	-	DeliveryFailed
getAPIKey	-	String	-
setAPIKey	String	-	-

14.4 Semantics

14.4.1 State Variables

APIKey: String

14.4.2 Environment Variables

connection: API connection

14.4.3 Assumptions

- Emailer API is operational and accessible
- ClinicalTrial.gov API is operational and accessible

14.4.4 Access Routine Semantics

sendEmail(emailAddress: String, subject: String, body: String):

- transition: None
- output: out := email request sent through the emailing API

- exception: exc := emailer returns error code → $DeliveryFailed$

getAPIKey():

- transition: None
- output: out := self.APIKey
- exception: None

setAPIKey(key: String):

- transition: self.APIKey := key
- output: None
- exception: None

14.4.5 Local Functions

findNewTrials():

- transition: None
- output: out := $\{trials : trial.postedDate > lastCheckedDate\}$
- exception: None

matchUsersToNewTrials():

- transition: None
- output: out := $\{user \times trial : user.conditions \subseteq trial.conditions\}$
- exception: None

15 MIS of Patient Data Collection Form Module

15.1 Module

PatientForm

15.2 Uses

InfoProfile, User

15.3 Syntax

15.3.1 Exported Constants

None

15.3.2 Exported Access Programs

Name	In	Out	Exceptions
displayPatientForm	-	ReactComponent	-

15.4 Semantics

15.4.1 State Variables

patientForm: ReactComponent
name: seq of char
dateOfBirth: seq of char
address: seq of char
gender: seq of char
healthDetails: TS Object<seq of char: Any>

15.4.2 Environment Variables

window
keyboard
mouse

15.4.3 Assumptions

None

15.4.4 Access Routine Semantics

displayPatientForm():

- transition: N/A
- output: out := self.patientForm
- exception: N/A

15.4.5 Local Functions

submitPatientDetails(newPatientDetails: TS Object<seq of char: Any>):

- transition: self.healthDetails = newPatientDetails
- output: N/A
- exception: $exc := length(newPatientDetails) == 0 \Rightarrow InvalidDetails$

16 MIS of User Login Form Module

16.1 Module

LoginForm

16.2 Uses

Login

16.3 Syntax

16.3.1 Exported Constants

None

16.3.2 Exported Access Programs

Name	In	Out	Exceptions
displayLoginForm	-	ReactComponent	-

16.4 Semantics

16.4.1 State Variables

loginForm: ReactComponent
emailAddress: seq of char
password: seq of char

16.4.2 Environment Variables

window
keyboard
mouse

16.4.3 Assumptions

None

16.4.4 Access Routine Semantics

displayLoginForm():

- transition: N/A
- output: out := self.loginForm
- exception: N/A

16.4.5 Local Functions

submitLoginDetails(newEmailAddress: seq of char, newPassword: seq of char):

- transition: self.emailAddress, self.password = newEmailAddress, newPassword
- output: N/A
- exception: $exc := length(newEmailAddress) == 0 \lor length(newPassword) == 0 \Rightarrow InvalidLoginDetails$

References

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL http://citeseer.ist.psu.edu/428727.html.

17 Appendix

[Extra information if required —SS]