

# 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

<b>1</b>	<b>Revision History</b>	<b>i</b>
<b>2</b>	<b>Symbols, Abbreviations and Acronyms</b>	<b>ii</b>
<b>3</b>	<b>Introduction</b>	<b>1</b>
<b>4</b>	<b>Notation</b>	<b>1</b>
<b>5</b>	<b>Module Decomposition</b>	<b>1</b>
<b>6</b>	<b>MIS of the User data module</b>	<b>3</b>
6.1	Module . . . . .	3
6.2	Uses . . . . .	3
6.3	Syntax . . . . .	3
6.3.1	Exported Constants . . . . .	3
6.3.2	Exported Access Programs . . . . .	3
6.4	Semantics . . . . .	3
6.4.1	State Variables . . . . .	3
6.4.2	Environment Variables . . . . .	3
6.4.3	Assumptions . . . . .	4
6.4.4	Access Routine Semantics . . . . .	4
6.4.5	Local Functions . . . . .	6
<b>7</b>	<b>MIS of Info Profile data module</b>	<b>6</b>
7.1	Module . . . . .	6
7.2	Uses . . . . .	6
7.3	Syntax . . . . .	6
7.3.1	Exported Constants . . . . .	6
7.3.2	Exported Access Programs . . . . .	6
7.4	Semantics . . . . .	7
7.4.1	State Variables . . . . .	7
7.4.2	Environment Variables . . . . .	7
7.4.3	Assumptions . . . . .	7
7.4.4	Access Routine Semantics . . . . .	7
7.4.5	Local Functions . . . . .	8
<b>8</b>	<b>MIS of Trial data module</b>	<b>9</b>
8.1	Module . . . . .	9
8.2	Uses . . . . .	9
8.3	Syntax . . . . .	9
8.3.1	Exported Constants . . . . .	9
8.3.2	Exported Access Programs . . . . .	9

8.4	Semantics . . . . .	9
8.4.1	State Variables . . . . .	9
8.4.2	Environment Variables . . . . .	9
8.4.3	Assumptions . . . . .	9
8.4.4	Access Routine Semantics . . . . .	9
8.4.5	Local Functions . . . . .	10
<b>9</b>	<b>MIS of the Fetch Trials Modules</b>	<b>11</b>
9.1	Module . . . . .	11
9.2	Uses . . . . .	11
9.3	Syntax . . . . .	11
9.3.1	Exported Constants . . . . .	11
9.3.2	Exported Access Programs . . . . .	11
9.4	Semantics . . . . .	11
9.4.1	State Variables . . . . .	11
9.4.2	Environment Variables . . . . .	11
9.4.3	Assumptions . . . . .	11
9.4.4	Access Routine Semantics . . . . .	12
9.4.5	Local Functions . . . . .	12
<b>10</b>	<b>MIS of the Trial Filtering Module</b>	<b>12</b>
10.1	Module . . . . .	12
10.2	Uses . . . . .	12
10.3	Syntax . . . . .	12
10.3.1	Exported Constants . . . . .	12
10.3.2	Exported Access Programs . . . . .	13
10.4	Semantics . . . . .	13
10.4.1	State Variables . . . . .	13
10.4.2	Environment Variables . . . . .	13
10.4.3	Assumptions . . . . .	13
10.4.4	Access Routine Semantics . . . . .	13
10.4.5	Local Functions . . . . .	14
<b>11</b>	<b>MIS of the Registration Module</b>	<b>15</b>
11.1	Module . . . . .	15
11.2	Uses . . . . .	15
11.3	Syntax . . . . .	15
11.3.1	Exported Constants . . . . .	15
11.3.2	Exported Access Programs . . . . .	15
11.4	Semantics . . . . .	15
11.4.1	State Variables . . . . .	15
11.4.2	Environment Variables . . . . .	15
11.4.3	Assumptions . . . . .	15

11.4.4	Access Routine Semantics . . . . .	15
11.4.5	Local Functions . . . . .	16
<b>12</b>	<b>MIS of the Login Module</b>	<b>16</b>
12.1	Module . . . . .	16
12.2	Uses . . . . .	16
12.3	Syntax . . . . .	16
12.3.1	Exported Constants . . . . .	16
12.3.2	Exported Access Programs . . . . .	16
12.4	Semantics . . . . .	16
12.4.1	State Variables . . . . .	16
12.4.2	Environment Variables . . . . .	16
12.4.3	Assumptions . . . . .	16
12.4.4	Access Routine Semantics . . . . .	16
12.4.5	Local Functions . . . . .	17
<b>13</b>	<b>MIS of the Email Template Module</b>	<b>17</b>
13.1	Module . . . . .	17
13.2	Uses . . . . .	17
13.3	Syntax . . . . .	17
13.3.1	Exported Constants . . . . .	17
13.3.2	Exported Access Programs . . . . .	17
13.4	Semantics . . . . .	17
13.4.1	State Variables . . . . .	17
13.4.2	Environment Variables . . . . .	17
13.4.3	Assumptions . . . . .	17
13.4.4	Access Routine Semantics . . . . .	17
13.4.5	Local Functions . . . . .	18
<b>14</b>	<b>MIS of the Email Notification Module</b>	<b>18</b>
14.1	Module . . . . .	18
14.2	Uses . . . . .	18
14.3	Syntax . . . . .	18
14.3.1	Exported Constants . . . . .	18
14.3.2	Exported Access Programs . . . . .	18
14.4	Semantics . . . . .	18
14.4.1	State Variables . . . . .	18
14.4.2	Environment Variables . . . . .	18
14.4.3	Assumptions . . . . .	18
14.4.4	Access Routine Semantics . . . . .	18
14.4.5	Local Functions . . . . .	19

<b>15 MIS of Patient Data Collection Form Module</b>	<b>19</b>
15.1 Module . . . . .	19
15.2 Uses . . . . .	19
15.3 Syntax . . . . .	20
15.3.1 Exported Constants . . . . .	20
15.3.2 Exported Access Programs . . . . .	20
15.4 Semantics . . . . .	20
15.4.1 State Variables . . . . .	20
15.4.2 Environment Variables . . . . .	20
15.4.3 Assumptions . . . . .	20
15.4.4 Access Routine Semantics . . . . .	20
15.4.5 Local Functions . . . . .	21
<b>16 MIS of User Login Form Module</b>	<b>21</b>
16.1 Module . . . . .	21
16.2 Uses . . . . .	21
16.3 Syntax . . . . .	21
16.3.1 Exported Constants . . . . .	21
16.3.2 Exported Access Programs . . . . .	21
16.4 Semantics . . . . .	21
16.4.1 State Variables . . . . .	21
16.4.2 Environment Variables . . . . .	21
16.4.3 Assumptions . . . . .	22
16.4.4 Access Routine Semantics . . . . .	22
16.4.5 Local Functions . . . . .	22
<b>17 MIS of User Profile Settings Module</b>	<b>22</b>
17.1 Module . . . . .	22
17.2 Uses . . . . .	22
17.3 Syntax . . . . .	22
17.3.1 Exported Constants . . . . .	22
17.3.2 Exported Access Programs . . . . .	22
17.4 Semantics . . . . .	23
17.4.1 State Variables . . . . .	23
17.4.2 Environment Variables . . . . .	23
17.4.3 Assumptions . . . . .	23
17.4.4 Access Routine Semantics . . . . .	23
17.4.5 Local Functions . . . . .	23
<b>18 Appendix</b>	<b>27</b>

### 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  $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | \dots | c_n \Rightarrow r_n)$ .

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 $(-\infty, \infty)$
natural number	$\mathbb{N}$	a number without a fractional component in $[1, \infty)$
real	$\mathbb{R}$	any number in $(-\infty, \infty)$

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:  $\text{out} := \text{self.name}$
- exception: N/A

setName(newName: seq of char):

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

getEmail():

- transition: N/A
- output:  $\text{out} := \text{self.email}$
- exception: N/A

setEmail(newEmail: seq of char):

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

getInfoProfiles():

- transition: N/A
- output:  $\text{out} := \text{self.infoProfiles}$
- exception: N/A

getInfoProfile(id: integer):

- transition: N/A

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

addInfoProfile(newInfoProfile: InfoProfile):

- transition:  $\text{self.infoProfiles} = \text{self.infoProfiles} + \text{newInfoProfile}$  (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:  $\text{self.infoProfiles} = \text{self.infoProfiles} - \text{oldInfoProfile}$  (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:  $\text{out} := \text{self.trials}$
- exception: N/A

getTrial(id: integer):

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

addTrial(newTrial: Trial):

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

removeTrial(oldTrial: Trial):

- transition:  $\text{self.trials} = \text{self.trials} - \text{oldTrial}$  (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:  $\text{out} := \text{DataFrame}$  populated with trials from the ClinicalTrials.gov API
- exception:  $\text{exc} := (\text{age} \notin (0, 120] \rightarrow \text{InvalidAge}) \vee (\neg \text{checkAddress}(\text{address}) \rightarrow \text{InvalidAddress}) \vee ((\exists x.x \in \text{parameters} : x = \varepsilon) \rightarrow \text{MissingParameter})$

#### 9.4.5 Local Functions

convertTrialsToDataFrame(rawData: csv):

- transition: None
- output:  $\text{out} := \text{rawData}$  formatted as a DataFrame
- exception: None

checkAddress(address: String):

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

## 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:  $\text{out} := \text{self.locator}$
- exception: None

setLocator(loc: geocoder):

- transition:  $\text{self.locator} = \text{loc}$
- output: None
- exception: None

#### 10.4.5 Local Functions

cleanAge(stringAge: String):

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

geodesicDistance(address: geocode, trialLocation: geocode):

- transition: None
- output:  $\text{out} := \arccos(\sin(\text{address.latitude}) \cdot \sin(\text{trialLocation.latitude}) + \cos(\text{address.latitude}) \cdot \cos(\text{trialLocation.latitude}) \cdot \cos(\text{trialLocation.longitude} - \text{address.longitude})) \cdot 6371000$
- exception: None

calculateDistance():

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

convertToJSON(df: DataFrame):

- transition: None
- output:  $\text{df} \rightarrow \text{json}(\text{df})$
- 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:  $\text{exc} := \text{emailer returns error code} \rightarrow \text{DeliveryFailed}$

getAPIKey():

- transition: None
- output:  $\text{out} := \text{self.APIKey}$
- exception: None

setAPIKey(key: String):

- transition:  $\text{self.APIKey} := \text{key}$
- output: None
- exception: None

#### 14.4.5 Local Functions

findNewTrials():

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

matchUsersToNewTrials():

- transition: None
- output:  $\text{out} := \{ \text{user} \times \text{trial} : \text{user.conditions} \subseteq \text{trial.conditions} \}$
- 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: String  
dateOfBirth: String  
address: String  
gender: String  
healthDetails: TS Object<String: 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<String: 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: String

password: String

#### 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: String, newPassword: String):

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

## 17 MIS of User Profile Settings Module

### 17.1 Module

ProfileSettings

### 17.2 Uses

User, InfoProfile, TrialFetcher, Trial

### 17.3 Syntax

#### 17.3.1 Exported Constants

None

#### 17.3.2 Exported Access Programs

Name	In	Out	Exceptions
displayProfileSettings	-	ReactComponent	-

## 17.4 Semantics

### 17.4.1 State Variables

profileSettings: ReactComponent  
name: String  
email: String  
savedTrials: list of Trial  
dateOfBirth: String  
address: String  
gender: String  
healthDetails: TS Object<String: Any>

### 17.4.2 Environment Variables

window  
keyboard  
mouse

### 17.4.3 Assumptions

None

### 17.4.4 Access Routine Semantics

displayProfileSettings():

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

### 17.4.5 Local Functions

getName():

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

setName(newName: String):

- 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: String):

- transition:  $self.email := newEmail$
- output: N/A
- exception:  $exc := length(newEmail) == 0 \Rightarrow InvalidEmail$

getSavedTrials():

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

removeTrial(oldTrial: Trial):

- transition:  $self.trials = \forall t \in self.trials \mid t \neq oldTrial$
- output: N/A
- exception: N/A

getDOB():

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

setDOB(newDOB: String):

- transition:  $self.dateOfBirth = newDOB$
- output: N/A
- exception:  $exc := length(newDOB) == 0 \Rightarrow EmptyDOB$

getAddress():

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

`setAddress(newAddress: String):`

- transition: `self.address = newAddress`
- output: N/A
- exception: *`exc := length(newAddress) == 0 ⇒ EmptyAddress`*

`getGender():`

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

`setGender(gender: String):`

- transition: `self.gender = gender`
- output: N/A
- exception: *`exc := length(gender) == 0 ⇒ EmptyGender`*

`getHealthDetails():`

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

`setHealthDetails(newHealthDetails: TS Object<String: Any>):`

- transition: `self.healthDetails = newHealthDetails`
- output: N/A
- exception: *`exc := length(newHealthDetails) == 0 ⇒ InvalidDetails`*



## 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>.

## 18 Appendix

[Extra information if required —SS]