Xiaoke Li(xl5u20)
Weiwu Qi(ww3u21)
Junhao Zhang(jz16u21)
Yangchen Kang(yk2g21)

COMP1216. Software Modelling and Design (2021-22)

Group 48: A COVID Vaccination Tracking System

Submission date: 13 05 2022

1 Introduction

Xiaoke Li(xl5u20) did the majority of this,like **first 2 machines and 2 context**. Weiwu Qi(ww3u21) did **UML**.Junhao Zhang(jz16u21) did **Booking parts**. Yangchen Kang(yk2g21) did **LaTex report and checking**. In conclusion, we made all those things together.And at each part of the whole work, anyone participated in any part.

2 Task 1. Class diagram

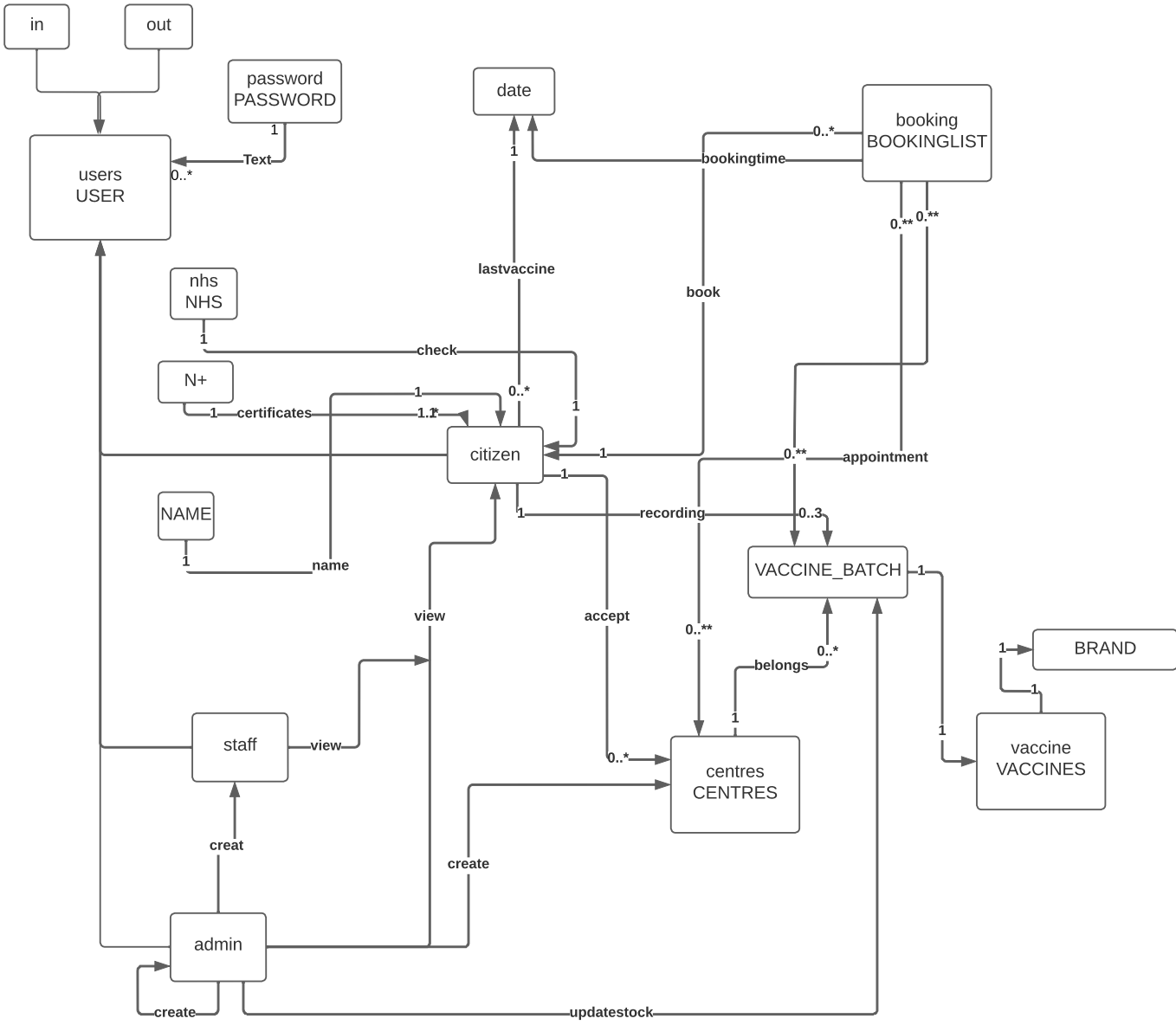

Figure 1: Class Diagram

3 Task 2. Event-B model

You are recommended to use the lstEventB.sty package for listing your Event-B model. We use the sample solution for Lab 7 (a hotel reception system) as an example here. The magic command is \EventBinputlisting to include any CamilleX source files (*.bucx, *.bumx) into your report.

```
context UserSystem
sets
 USERS
 PASSWORD
constants
 root
 default_password
axioms
 @def1: root ∈ USERS
 @def2: default_password ∈ PASSWORD
end
```

```
machine Users
sees UserSystem
variables
 users in out account
 adminstrators
 staff
 citizens

invariants
 @inv1: users ⊆ USERS
 @inv2: adminstrators ⊆ users
 @inv3: staff ⊆ users
 @inv4: citizens ⊆ users
 @inv5: partition(users,adminstrators,staff,citizens)
 @inv6: partition(users, in ,out)
 // Each user has exactly 1 PASSWORD
 //A PASSWORD may be used by many users
 @inv7: account ∈ users → PASSWORD // total fucntion

events
 event INITIALISATION
 then
  @act1: users := {root}  // root administrator
  @act2: adminstrators := {root}
  @act3: out := {root}
  @act4: account := {root ↦ default_password}
  @act5: staff,citizens,in := ∅,∅,∅
 end

 /*
  * Let registered users to login in usin password
  */
 event Login
 any u p where
```

```
35   @grd1: u ∈ users
36   @grd2: u ∉ in
37   @grd3: p ∈ PASSWORD
38   @grd4: account(u) = p // user and PASSWORD must match, then could login
39  then
40   @act1: in := in ∪ {u}
41   @act2: out := out \{u}
42  end
43
44  /*
45   * Let registered users to login out
46   **/
47  event Logout
48  any s where
49   @grd1: s ∈ in
50  then
51   @act1: in := in \ {s}
52   @act2: out := out ∪ {s}
53  end
54
55  /*
56   * Users could change their password
57   */
58  event ChangePassword
59  any p u  where
60   @grd1: u ∈ users
61   @grd2: p ∈ PASSWORD
62   @grd3: account(u) ≠ p // New PASSWORD is not the  current PASSWORD
63  then
64   @act1: account(u) := p
65  end
66
67  /*
68   * Citizen could register by their own
69   */
70  event RegisterCitizen
71  any c p where
72   @grd1: c ∉ users
73   @grd2: p ∈ PASSWORD
74  then
75   @act1: users := users ∪ {c}
76   @act2: citizens :=  citizens ∪ {c}
77   @act3: out := out ∪ {c}
78   @act4: account(c) := p
79  end
80
81  /*
82   * Staff must be registered by a administrator
83   */
84  event RegisterStaff
85  any a1 s  p where
86   @grd1: a1 ∈ adminstrators ∩ in
87   @grd4: s ∈ (USERS \ users)
88   @grd5: p ∈ PASSWORD
89  then
90   @act1: users := users ∪ {s}
```

```
    @act2: staff := staff ∪ {s}
    @act3: account(s) := p
    @act4: out := out ∪ {s}
  end

  /*
   * administrator must be registered by another administrator
   **/
  event RegisterAd
  any a1 a2 p where
    @grd1: a1 ∈ adminstrators ∩ in
    @grd2: a2 ∈ (USERS \ users)
    @grd3: p ∈ PASSWORD
  then
    @act1: users := users ∪ {a2}
    @act2: adminstrators := adminstrators ∪ {a2}
    @act3: account(a2) := p
    @act4: out := out ∪ {a2}
  end

end
```

```
context VaccinationSystem
sets
CENTRES // Sets of vaccination centres
BRAND//BRAND
VACCINES_BATCH  // The set of vaccines  numbers
VACCINES

NAME
NHS
end
```

```
machine VaccinationCentre
refines Users
sees VaccinationSystem UserSystem
variables
 users in out account adminstrators staff citizens
 date
 nhs // sets of existing NHS number
 name//name of citizen
 check // Each citizne has unique nhs
 recording//recording of vaccines
 certificates // total recieved vaccines times
 lastVaccine //last time get vaccine

 centres //Existing centres
 vaccines //Sets of vaccines
 brand// relation of brands and vaccine
 batch// relation of vaccin and its batch
 belongs // All the centres have vaccines(at least 1)
invariants
```

```
  @inv10: date ⊆ ℕ1
  @inv11: name ∈ citizens → NAME
  @inv12: nhs ⊆ NHS
  @inv13: check ∈ citizens ⤖ nhs
  @inv14: recording ∈ citizens ↔ VACCINES_BATCH //recording of vaccine
  @inv15: certificates ∈ citizens → ℤ //numbers of vaccines had
  @inv16: lastVaccine ∈ citizens ⇸ date //partiacal

  @inv99: vaccines ⊆ VACCINES
  @inv17: centres ⊆ CENTRES
  @inv18: brand ∈ vaccines → BRAND
  @inv19: belongs ∈ centres ↔ VACCINES_BATCH //relation, a centre may runs out of stock
  @inv20: batch ∈ VACCINES_BATCH ⤖ vaccines //vaccines has 1 exaclty batch

  @grd1: ∀c·c ∈ citizens ⇒ certificates(c) ≤ 3
events
 event INITIALISATION extends INITIALISATION
 then
  @act6: name,nhs,recording,certificates,lastVaccine,check := ∅,∅,∅,∅,∅,∅
  @act7: date := ∅
  @act8: vaccines,centres,belongs,brand,batch := ∅,∅,∅,∅,∅
 end

 /*
  * A centre must be added by a administrator
  */
 event AddCentre
 any a c ba where
  @grd1: a ∈ adminstrators ∩ in
  @grd2: c ∉ centres
  @grd3: ba ∈ dom(batch) //@ba is existing vaccine number
  @grd4: ba ∉ ran(belongs) //@ba has not in any centre
 then
  @act1: centres := centres ∪ {c}
  @act2: belongs(c) := ba
 end

 /*
  * A stock must be added by a administrator
  */
 event UpdateStock
 any a c ba  where
  @grd1: a ∈ adminstrators ∩ in
  @grd2: c ∈ centres
  @grd3: ba ⊆ VACCINES_BATCH
  @grd4: ba ⊆ dom(batch) //@ba is existing vaccine number
  @grd5: ba ∩ ran(belongs) = ∅ //@ba has not in any centre
 then
  @act1: belongs := belongs ∪ ({c} × ba ) //overradding
 end

 /*
  * Add new vaccine with its batch
  */
 event NewVaccine//matching vaccine and batch
 any ba v br where
```

```
76    @grd1: v ∉ vaccines
77    @grd2: ba ∈ VACCINES_BATCH
78    @grd3: br ∈ BRAND
79    @grd4: ba ∉ dom(batch)
80   then
81    @act1: vaccines := vaccines ∪ {v}
82    @act2: brand(v) := br
83    @act3: batch(ba) := v
84   end
85
86  /*
87  * Extend RegisterCitizen:
88  * a citizen must has a name and its nhs number
89  */
90   event RegisterCitizen extends RegisterCitizen //to bind name∧NHS
91   any n1 n2 where
92    @grd3: n1 ∈ NAME
93    @grd4: n2 ∈ NHS \ nhs
94   then
95    @act5: nhs := nhs ∪ {n2}
96    @act6: check(c) := n2
97    @act7: name(c) := n1
98    @act8: certificates(c) := 0
99    end
100
101  /**
102   * An adiministrator or staff could view his detail
103   */
104  event ViewCitizenDetail
105  any a num rs where
106   @grd1: a ∈ (adminstrators ∪ staff) ∩ in
107   @grd2: num ∈ nhs
108   @grd3: rs = check∼(num) //get citizen details by @num
109  end
110
111  /* An adiministrator or staff could update the citizen's certificates,
112   * and change the stock
113   * */
114  event UpdateCertificates
115  any a c1 c2 v1 t where
116   @grd1: a ∈ (adminstrators ∪ staff) ∩ in
117   @grd2: c1 ∈ citizens
118   @grd3: c2 ∈ centres
119   @grd4: v1 ∈ VACCINES_BATCH
120   @grd5: t ∈ date
121   @grd7: {c2 ↦ v1 } ⊂ belongs //vaccine is in @c2 center
122   @grd8: certificates(c1) + 1 ≤ 3
123  then
124   @act1: recording := {c1 ↦ v1} ∪ recording
125   @act2: lastVaccine(c1) := t
126   @act3: certificates(c1) := certificates(c1) +1
127   @act4: belongs := belongs ⩥ {v1}
128  end
129
130  /**
131   * An citizen could view own detail
```

```
 */
event ViewMyCertificates
any c rs1 rs2 where
  @grd1: c ∈ citizens ∩ in
  @grd2: rs1 = recording[{c}] // return the recoding of @c
  @grd3: rs2 = certificates(c)
end

event RegisterStaff extends RegisterStaff
end

event RegisterAd extends RegisterAd
end

event Login extends Login
end

event Logout extends Logout
end

event ChangePassword extends ChangePassword
end

end
```

```
context BookingSystem
extends VaccinationSystem
sets
BOOKINGLIST
end
```

```
machine Booking
refines VaccinationCentre
sees BookingSystem UserSystem
variables
 users
 in
 out
 account
 adminstrators
 staff
 citizens
 date
 nhs
 name
 check
 recording
 certificates
 lastVaccine
 centres
 vaccines
 brand
```

```
22  batch
23  belongs
24
25  booking  // recorgin fo booking
26  book//the book process of citizens
27  appointment//the centres is avialbe for booking
28  bookingtime//availbe time for booking
29  accept//the citizen accept the booking and its centres
30 invariants
31  @inv31: booking ⊆ BOOKINGLIST
32  @inv32: book ∈ booking → citizens //citizen could have a booking
33  @inv33: appointment ∈ centres ⇸ booking
34  @inv34: bookingtime ∈ booking ↔ ℕ1
35  @inv35: accept ∈ citizens → centres
36 events
37  event INITIALISATION extends INITIALISATION
38  then
39   @act31: booking,book,appointment,bookingtime,accept := ∅,∅,∅,∅,∅
40  end
41
42  /*
43   * To create a booking process
44   * a citizen will be given a availbe booking,
45   * containg center time booking.
46   *
47   **/
48  event NewBooking
49  any c1 c2 b t where
50   @grd1: c1 ∈ citizens
51   @grd2: c2 ∈ centres
52   @grd3: b ∈ booking
53   @grd4: t ∈ ℕ1
54   @grd5: (c2 ↦ b) ∈ appointment
55   @grd6: (b ↦ t) ∈ bookingtime
56   @grd7: certificates(c1) ≤ 3
57   @grd8: t − lastVaccine(c1) ≥ 28
58  then
59   @act1: book(b) := c1
60  end
61
62  /*A citizen accept the booking, and he or she will be turn in the recoring of which centre */
63  event CitizenAcceptBooking
64  any b c1 c2 where
65   @grd1: b ∈ booking
66   @grd2: c2 ∈ centres
67   @grd3: c1 ∈ citizens
68   @grd5: (c2 ↦ b) ∈ appointment
69   @grd6: accept(c1) ≠ c2
70  then
71   @act1: accept(c1) := c2
72  end
73
74  /* A citizen could reject booking and he or she would be given an new booking*/
75 event CitizenRejectBooking
76 any b c1 c2  b3 c3 where
77   @grd1: b ∈ booking
```

```
78  @grd2: c2 ∈ centres
79  @grd3: c1 ∈ citizens
80  @grd5: (c2 ↦ b) ∈ appointment
81  @grd6: accept(c1) ≠ c2
82  @grd7: b3 ∈ dom(bookingtime)
83  @grd9: (c3 ↦ b3) ∉ appointment
84  @grd10: book(b) = c1
85 then
86  @act1: book := book ⩥ {c1}
87  @act2: appointment := appointment ∪ ({c3 }× {b3})
88  @act3: book(b3) := c1
89 end
90
91 /* Citizen terminate booking process, and would be out of the booking process */
92  event CitizenTerminateABooking
93  any c1 b c2 where
94   @grd1: c1 ∈ citizens
95   @grd2: (b ↦ c1 ) ∈ book
96   @grd3: (c1 ↦ c2) ∈ accept
97  then
98   @act1: book := book ⩥ {c1}
99   @act2: accept :={c1} ⩤ accept
100 end
101
102  /** Return which centre he is in */
103 event ViewCurrentBooking
104 any c1  rs1  where
105  @grd1: c1 ∈ citizens
106  @grd2: c1 ∈ dom(accept)
107  @grd3: rs1 = accept(c1)
108 end
109
110  /** Adding the appointment with availble dates and centres*/
111 event UpdateAppointment
112 any b c t where
113  @grd1: b ⊆ booking
114  @grd2: c ∈ centres
115  @grd3: t ∈ ℕ1
116 then
117  @act1: appointment := appointment ∪ (centres × b)
118  @act2: bookingtime := (b × {t} )∪ bookingtime
119 end
120
121 event AddCentre extends AddCentre
122 any b where
123  @grd31: b ⊆ booking
124 then
125  @act31: appointment := appointment ∪ (centres × b)
126 end
127
128 event UpdateStock extends UpdateStock
129 end
130
131 event NewVaccine extends NewVaccine
132 end
133
```

```
134   event RegisterCitizen extends RegisterCitizen
135   end
136
137   event ViewCitizenDetail extends ViewCitizenDetail
138   end
139
140   event UpdateCertificates extends UpdateCertificates
141   end
142
143   event ViewMyCertificates extends ViewMyCertificates
144   end
145
146   event RegisterStaff extends RegisterStaff
147   end
148
149   event RegisterAd extends RegisterAd
150   end
151
152   event Login extends Login
153   end
154
155   event Logout extends Logout
156   end
157
158   event ChangePassword extends ChangePassword
159   end
160
161 end
```