section *homestay* **parents** *standard_toolkit*

Contents

1 General

We want to have a few basic types here. We need to know general information about all of the users in the system:

- Email address, this is the unique identifier in the system

- Password, this has restrictions:

  - At least 8 characters in length
  - 1 digit
  - 1 uppercase character
  - 1 lowercase character

- First name, should be non-empty

- Last name, should be non-empty

[*Email*
, *Password*
, *FirstName*
, *LastName*
, *StreetNumber*
, *StreetName*
, *City*
, *State*
, *ZipCode*
, *Phone*
, *Photo*
, *Text*]

We create a couple of synonyms for photo types.

PostingPhoto == *Photo*
UserPhoto == *Photo*

We're going to construct a date type.

2

$$
\begin{aligned}
Month ::=\ &January\\
|\ &February\\
|\ &March\\
|\ &April\\
|\ &May\\
|\ &June\\
|\ &July\\
|\ &August\\
|\ &September\\
|\ &October\\
|\ &November\\
|\ &December\\
Day ==\ &\mathbb{N}\\
Year ==\ &\mathbb{N}\\
Date ==\ &Month \times Day \times Year
\end{aligned}
$$

We'll need a couple of flags.

We want to know whether or not the user is an admin or not. This is used so that they get access to certain features in the application.

$$
\begin{aligned}
AdminFlag ::=\ &\\
&Admin\\
|\ &NotAdmin\\
InappropriateFlag ::=\ &\\
&Zero\\
|\ &First\\
|\ &Second\\
|\ &Third\\
|\ &Remove
\end{aligned}
$$

We need to know about the host stuff.

Smoking ::=
   EnjoysSmoking
   | *NonSmoking*
Pets ::=
   NoPets
   | *YesPets*
Children ::=
   One
   | *Two*
   | *ThreePlus*
Diet ::=
   GlutenFree
   | *Omnivore*
   | *Pescatarian*
   | *Vegetarian*
   | *Vegan*
   | *OtherDiet*
Religion ::=
   Agnostic
   | *Athiest*
   | *Buddhist*
   | *Christian*
   | *Catholic*
   | *Mormon*
   | *Muslim*
   | *OtherReligion*
ExactOrArea ::=
   ExactLocation
   | *Area*

We need to know how the two people feel about each other.

MatchFlag ::=
   Accept
   | *Reject*

Using these basic types we construct more complex types in the system. The applicant is the user of the system. Password tokens are used to reset the password.

$Applicant == Email \times Password \times FirstName \times LastName \times AdminFlag$
$PToken == \mathbb{N} \times Email$
$IFlag == Email \times InappropriateFlag$
$Message == (Applicant \times Applicant) \times Text$

Now want to create some stuff specific to the host, but that needs to be used by other parts of the application.

$Price == \mathbb{N}$
$SizeOfRoom == \mathbb{N} \times \mathbb{N}$
$Address == StreetNumber \times StreetName \times City \times State \times ZipCode$
$AddressInfo == Address \times ExactOrArea$
$Availability == Date \times Date$
$HostPreference == Applicant \times Smoking \times Pets \times Children \times Diet \times Religion$
$Posting == HostPreference \times AddressInfo \times SizeOfRoom \times Price \times Availability \times PostingPhoto$

We need to be able to notify the user of what is happening in the application. So, we enumerate the possible responses from actions taken.

$Response ::=$
    $InvalidToken$
    $|\ PasswordResetSuccessful$
    $|\ LoginSuccessful$
    $|\ InvalidAvailability$
    $|\ ValidAvailability$

1.1 Application State

Our initial state is just a bunch of empty sets.

```
┌─ HomestayInitial ──────────────────────────────────
│ Applicants : ℙ Applicant
│ Emails : ℙ Email
│ Passwords : ℙ Password
│ PhoneNumbers : ℙ(Applicant × Phone)
│ UserPhotos : ℙ(Applicant × Photo)
│ Valid, Invalid : ℙ PToken
│ IFlags : ℙ IFlag
│ Postings : ℙ Posting
│ HostPreferences : ℙ HostPreference
│ Chats : ℙ Message
│ Matches : ℙ(Applicant × Applicant × MatchFlag)
├────────────────────────────────────────────────────
│ Applicants = ∅
│ Chats = ∅
│ Emails = ∅
│ HostPreferences = ∅
│ IFlags = ∅
│ Matches = ∅
│ Passwords = ∅
│ PhoneNumbers = ∅
│ Postings = ∅
│ UserPhotos = ∅
│ Valid = Invalid = ∅
└────────────────────────────────────────────────────
```

In the database we need to ensure that there is each email address is unique, and that the password tokens are either valid or invalid.

```
┌─ HomestayDatabase ─────────────────────────────────
│ HomestayInitial
├────────────────────────────────────────────────────
│ ∀ e_0, e_1 : Email; p_0, p_1 : Password; fn_0, fn_1 : FirstName; ln_0, ln_1 : LastName •
│     e_0 = e_1 ⇒ (p_0 = p_1 ∧ fn_0 = fn_1 ∧ ln_0 = ln_1)
│ Valid ∩ Invalid = ∅
└────────────────────────────────────────────────────
```

We want an easy way to update an applicant. This helper function updates the applicant in the set of all applicants, identified by the email address.

function 42 **leftassoc** ($\_ \otimes \_$)

$$\_ \otimes \_ : \mathbb{P}\,Applicant \times Applicant \rightarrow \mathbb{P}\,Applicant$$

$$\forall\, a_1 : Applicant;\ as : \mathbb{P}\,Applicant\ \bullet$$
$$\exists\, p_0, p_1 : Password;$$
$$fn_0, fn_1 : FirstName;$$
$$ln_0, ln_1 : LastName;$$
$$ad_0, ad_1 : AdminFlag;$$
$$e : Email;$$
$$a_0 : Applicant\ |$$
$$a_1 = (e, p_1, fn_1, ln_1, ad_1) \wedge (a_0 = (e, p_0, fn_0, ln_0, ad_0) \in as)\ \bullet$$
$$as \otimes a_1 = (as \setminus \{a_0\}) \cup \{a_1\}$$

1.2 General Operators

It gets kind of silly to have to rewrite these operators and functions each time, so we have some helpers here.

1.2.1 General Projections

section *projections* **parents** *homestay*

$[A, B, C]$
$$firstOf3 == \lambda\, a : A;\ b : B;\ c : C \bullet a$$
$$secondOf3 == \lambda\, a : A;\ b : B;\ c : C \bullet b$$
$$thirdOf3 == \lambda\, a : A;\ b : B;\ c : C \bullet c$$

$[A, B, C, D]$
$$firstOf4 == \lambda\, a : A;\ b : B;\ c : C;\ d : D \bullet a$$
$$secondOf4 == \lambda\, a : A;\ b : B;\ c : C;\ d : D \bullet b$$
$$thirdOf4 == \lambda\, a : A;\ b : B;\ c : C;\ d : D \bullet c$$
$$fourthOf4 == \lambda\, a : A;\ b : B;\ c : C;\ d : D \bullet d$$

$[A, B, C, D, E]$
$$firstOf5 == \lambda\, a : A;\ b : B;\ c : C;\ d : D;\ e : E \bullet a$$
$$secondOf5 == \lambda\, a : A;\ b : B;\ c : C;\ d : D;\ e : E \bullet b$$
$$thirdOf5 == \lambda\, a : A;\ b : B;\ c : C;\ d : D;\ e : E \bullet c$$
$$fourthOf5 == \lambda\, a : A;\ b : B;\ c : C;\ d : D;\ e : E \bullet d$$
$$fifthOf5 == \lambda\, a : A;\ b : B;\ c : C;\ d : D;\ e : E \bullet e$$

```
┌─[A, B, C, D, E, F]────────────────────────────────
│  firstOf6 == λ a : A;  b : B;  c : C;  d : D;  e : E;  f : F • a
│  secondOf6 == λ a : A;  b : B;  c : C;  d : D;  e : E;  f : F • b
│  thirdOf6 == λ a : A;  b : B;  c : C;  d : D;  e : E;  f : F • c
│  fourthOf6 == λ a : A;  b : B;  c : C;  d : D;  e : E;  f : F • d
│  fifthOf6 == λ a : A;  b : B;  c : C;  d : D;  e : E;  f : F • e
│  sixthOf6 == λ a : A;  b : B;  c : C;  d : D;  e : E;  f : F • f
└──────────────────────────────────────────────────
```

1.2.2 General Updates

section *updates* **parents** *homestay*

Updating for 3-tuples.

```
┌─[A, B, C]─────────────────────────────────
│  updateFirstOf3 : (A × B × C) × A→
│                   A × B × C
│ ───────────────────────────
│  ∀ abc : A × B × C;  a : A •
│      ∃ a₀ : A;  b : B;  c : C | abc = (a₀, b, c) •
│          updateFirstOf3(abc, a) = (a, b, c)
└───────────────────────────────────────────
```

```
┌─[A, B, C]─────────────────────────────────
│  updateSecondOf3 : (A × B × C) × B→
│                    A × B × C
│ ───────────────────────────
│  ∀ abc : A × B × C;  b : B •
│      ∃ a : A;  b₀ : B;  c : C | abc = (a, b₀, c) •
│          updateSecondOf3(abc, b) = (a, b, c)
└───────────────────────────────────────────
```

```
┌─[A, B, C]─────────────────────────────────
│  updateThirdOf3 : (A × B × C) × C→
│                   A × B × C
│ ───────────────────────────
│  ∀ abc : A × B × C;  c : C •
│      ∃ a : A;  b : B;  c₀ : C | abc = (a, b, c₀) •
│          updateThirdOf3(abc, c) = (a, b, c)
└───────────────────────────────────────────
```

Updating for 4-tuples.

8

$\boxed{[A, B, C, D]}$
$updateFirstOf4 : (A \times B \times C \times D) \times A \to$
$\qquad A \times B \times C \times D$

$\forall\, abcd : A \times B \times C \times D;\ a : A \bullet$
$\quad \exists\, a_0 : A;\ b : B;\ c : C;\ d : D \mid abcd = (a_0, b, c, d) \bullet$
$\qquad updateFirstOf4(abcd, a) = (a, b, c, d)$

$\boxed{[A, B, C, D]}$
$updateSecondOf4 : (A \times B \times C \times D) \times B \to$
$\qquad A \times B \times C \times D$

$\forall\, abcd : A \times B \times C \times D;\ b : B \bullet$
$\quad \exists\, a : A;\ b_0 : B;\ c : C;\ d : D \mid abcd = (a, b_0, c, d) \bullet$
$\qquad updateSecondOf4(abcd, b) = (a, b, c, d)$

$\boxed{[A, B, C, D]}$
$updateThirdOf4 : (A \times B \times C \times D) \times C \to$
$\qquad A \times B \times C \times D$

$\forall\, abcd : A \times B \times C \times D;\ c : C \bullet$
$\quad \exists\, a : A;\ b : B;\ c_0 : C;\ d : D \mid abcd = (a, b, c_0, d) \bullet$
$\qquad updateThirdOf4(abcd, c) = (a, b, c, d)$

$\boxed{[A, B, C, D]}$
$updateFourthOf4 : (A \times B \times C \times D) \times D \to$
$\qquad A \times B \times C \times D$

$\forall\, abcd : A \times B \times C \times D;\ d : D \bullet$
$\quad \exists\, a : A;\ b : B;\ c : C;\ d_0 : D \mid abcd = (a, b, c, d_0) \bullet$
$\qquad updateFourthOf4(abcd, d) = (a, b, c, d)$

Updating for 5-tuples.

$\boxed{[A, B, C, D, E]}$
$updateFirstOf5 : (A \times B \times C \times D \times E) \times A \to$
$\qquad A \times B \times C \times D \times E$

$\forall\, abcde : A \times B \times C \times D \times E;\ a : A \bullet$
$\quad \exists\, a_0 : A;\ b : B;\ c : C;\ d : D;\ e : E \mid abcde = (a_0, b, c, d, e) \bullet$
$\qquad updateFirstOf5(abcde, a) = (a, b, c, d, e)$

9

$\quad$ $[A, B, C, D, E]$ $\quad\quad\quad\quad\quad\quad\quad\quad\quad$
$\quad updateSecondOf\,5 : (A \times B \times C \times D \times E) \times B \rightarrow$
$\quad\quad\quad\quad\quad\quad A \times B \times C \times D \times E$

$\quad \forall\, abcde : A \times B \times C \times D \times E; \ b : B \ \bullet$
$\quad\quad \exists\, a : A; \ b_0 : B; \ c : C; \ d : D; \ e : E \mid abcde = (a, b_0, c, d, e) \ \bullet$
$\quad\quad\quad updateSecondOf\,5(abcde, b) = (a, b, c, d, e)$

$\quad$ $[A, B, C, D, E]$ $\quad\quad\quad\quad\quad\quad\quad\quad\quad$
$\quad updateThirdOf\,5 : (A \times B \times C \times D \times E) \times C \rightarrow$
$\quad\quad\quad\quad\quad\quad A \times B \times C \times D \times E$

$\quad \forall\, abcde : A \times B \times C \times D \times E; \ c : C \ \bullet$
$\quad\quad \exists\, a : A; \ b : B; \ c_0 : C; \ d : D; \ e : E \mid abcde = (a, b, c_0, d, e) \ \bullet$
$\quad\quad\quad updateThirdOf\,5(abcde, c) = (a, b, c, d, e)$

$\quad$ $[A, B, C, D, E]$ $\quad\quad\quad\quad\quad\quad\quad\quad\quad$
$\quad updateFourthOf\,5 : (A \times B \times C \times D \times E) \times D \rightarrow$
$\quad\quad\quad\quad\quad\quad A \times B \times C \times D \times E$

$\quad \forall\, abcde : A \times B \times C \times D \times E; \ d : D \ \bullet$
$\quad\quad \exists\, a : A; \ b : B; \ c : C; \ d_0 : D; \ e : E \mid abcde = (a, b, c, d_0, e) \ \bullet$
$\quad\quad\quad updateFourthOf\,5(abcde, d) = (a, b, c, d, e)$

$\quad$ $[A, B, C, D, E]$ $\quad\quad\quad\quad\quad\quad\quad\quad\quad$
$\quad updateFifthOf\,5 : (A \times B \times C \times D \times E) \times E \rightarrow$
$\quad\quad\quad\quad\quad\quad A \times B \times C \times D \times E$

$\quad \forall\, abcde : A \times B \times C \times D \times E; \ e : E \ \bullet$
$\quad\quad \exists\, a : A; \ b : B; \ c : C; \ d : D; \ e_0 : E \mid abcde = (a, b, c, d, e_0) \ \bullet$
$\quad\quad\quad updateFifthOf\,5(abcde, e) = (a, b, c, d, e)$

Updating for 6-tuples.

$\quad$ $[A, B, C, D, E, F]$ $\quad\quad\quad\quad\quad\quad\quad\quad$
$\quad updateFirstOf\,6 : (A \times B \times C \times D \times E \times F) \times A \rightarrow$
$\quad\quad\quad\quad\quad\quad A \times B \times C \times D \times E \times F$

$\quad \forall\, abcdef : A \times B \times C \times D \times E \times F; \ a : A \ \bullet$
$\quad\quad \exists\, a_0 : A; \ b : B; \ c : C; \ d : D; \ e : E; \ f : F \mid abcdef = (a_0, b, c, d, e, f) \ \bullet$
$\quad\quad\quad updateFirstOf\,6(abcdef, a) = (a, b, c, d, e, f)$

10

$\boxed{[A, B, C, D, E, F]}$
$updateSecondOf6 : (A \times B \times C \times D \times E \times F) \times B \rightarrow$
$\qquad\qquad A \times B \times C \times D \times E \times F$

$\forall\, abcdef : A \times B \times C \times D \times E \times F;\ b : B \bullet$
$\quad \exists\, a : A;\ b_0 : B;\ c : C;\ d : D;\ e : E;\ f : F \mid abcdef = (a, b_0, c, d, e, f) \bullet$
$\qquad updateSecondOf6(abcdef, b) = (a, b, c, d, e, f)$

$\boxed{[A, B, C, D, E, F]}$
$updateThirdOf6 : (A \times B \times C \times D \times E \times F) \times C \rightarrow$
$\qquad\qquad A \times B \times C \times D \times E \times F$

$\forall\, abcdef : A \times B \times C \times D \times E \times F;\ c : C \bullet$
$\quad \exists\, a : A;\ b : B;\ c_0 : C;\ d : D;\ e : E;\ f : F \mid abcdef = (a, b, c_0, d, e, f) \bullet$
$\qquad updateThirdOf6(abcdef, c) = (a, b, c, d, e, f)$

$\boxed{[A, B, C, D, E, F]}$
$updateFourthOf6 : (A \times B \times C \times D \times E \times F) \times D \rightarrow$
$\qquad\qquad A \times B \times C \times D \times E \times F$

$\forall\, abcdef : A \times B \times C \times D \times E \times F;\ d : D \bullet$
$\quad \exists\, a : A;\ b : B;\ c : C;\ d_0 : D;\ e : E;\ f : F \mid abcdef = (a, b, c, d_0, e, f) \bullet$
$\qquad updateFourthOf6(abcdef, d) = (a, b, c, d, e, f)$

$\boxed{[A, B, C, D, E, F]}$
$updateFifthOf6 : (A \times B \times C \times D \times E \times F) \times E \rightarrow$
$\qquad\qquad A \times B \times C \times D \times E \times F$

$\forall\, abcdef : A \times B \times C \times D \times E \times F;\ e : E \bullet$
$\quad \exists\, a : A;\ b : B;\ c : C;\ d : D;\ e_0 : E;\ f : F \mid abcdef = (a, b, c, d, e_0, f) \bullet$
$\qquad updateFifthOf6(abcdef, e) = (a, b, c, d, e, f)$

$\boxed{[A, B, C, D, E, F]}$
$updateSixthOf6 : (A \times B \times C \times D \times E \times F) \times F \rightarrow$
$\qquad\qquad A \times B \times C \times D \times E \times F$

$\forall\, abcdef : A \times B \times C \times D \times E \times F;\ f : F \bullet$
$\quad \exists\, a : A;\ b : B;\ c : C;\ d : D;\ e : E;\ f_0 : F \mid abcdef = (a, b, c, d, e, f_0) \bullet$
$\qquad updateSixthOf6(abcdef, f) = (a, b, c, d, e, f)$

11

1.3 Morphisms

1.3.1 Month

$month2Nat : Month \rightarrow \mathbb{N}$

$month2Nat\ January = 1\ \wedge$
$month2Nat\ February = 2\ \wedge$
$month2Nat\ March = 3\ \wedge$
$month2Nat\ April = 4\ \wedge$
$month2Nat\ May = 5\ \wedge$
$month2Nat\ June = 6\ \wedge$
$month2Nat\ July = 7\ \wedge$
$month2Nat\ August = 8\ \wedge$
$month2Nat\ September = 9\ \wedge$
$month2Nat\ October = 10\ \wedge$
$month2Nat\ November = 11\ \wedge$
$month2Nat\ December = 12$

$nat2Month : \mathbb{N} \nrightarrow Month$

$nat2Month\ 1 = January\ \wedge$
$nat2Month\ 2 = February\ \wedge$
$nat2Month\ 3 = March\ \wedge$
$nat2Month\ 4 = April\ \wedge$
$nat2Month\ 5 = May\ \wedge$
$nat2Month\ 6 = June\ \wedge$
$nat2Month\ 7 = July\ \wedge$
$nat2Month\ 8 = August\ \wedge$
$nat2Month\ 9 = September\ \wedge$
$nat2Month\ 10 = October\ \wedge$
$nat2Month\ 11 = November\ \wedge$
$nat2Month\ 12 = December$

1.3.2 InappropriateFlag

$InFlag2Nat : InappropriateFlag \rightarrow \mathbb{N}$

$InFlag2Nat\ Zero = 0\ \wedge$
$InFlag2Nat\ First = 1\ \wedge$
$InFlag2Nat\ Second = 2\ \wedge$
$InFlag2Nat\ Third = 3\ \wedge$
$InFlag2Nat\ Remove = 4$

$Nat2InFlag : \mathbb{N} \nrightarrow InappropriateFlag$

$Nat2InFlag\ 0 = Zero\ \wedge$
$Nat2InFlag\ 1 = First\ \wedge$
$Nat2InFlag\ 2 = Second\ \wedge$
$Nat2InFlag\ 3 = Third\ \wedge$
$Nat2InFlag\ 4 = Remove$

2 Login

When we go to create a new account, we need some information from the user. We update all of our sets to reflect the new addition. After they have successfully created an account, they are taken to the main menu.

CreateUserAccount

$\Delta HomestayDatabase$
$E? : Email$
$FN? : FirstName$
$LN? : LastName$
$P? : Password$

$E? \notin Emails$
$Emails' = Emails \cup \{E?\}$
$Passwords' = Passwords \cup \{P?\}$
$Applicants' = Applicants \cup \{(E?, P?, FN?, LN?, NotAdmin)\}$
$PhoneNumbers' = PhoneNumbers$
$UserPhoto' = UserPhoto$
$Valid' = Valid$
$Invalid' = Invalid$
$Postings' = Postings$
$HostPreferences' = HostPreferences$

```
┌─ CreateAdminAccount ─────────────────────────────────┐
│ ΔHomestayDatabase                                     │
│ E? : Email                                            │
│ FN? : FirstName                                       │
│ LN? : LastName                                        │
│ P? : Password                                         │
│ AD? : AdminFlag                                       │
├───────────────────────────────────────────────────────┤
│ E? ∉ Emails                                           │
│ AD? = Admin                                           │
│ Emails' = Emails ∪ {E?}                              │
│ Passwords' = Passwords ∪ {P?}                        │
│ Applicants' = Applicants ∪ {(E?, P?, FN?, LN?, AD?)} │
│ PhoneNumbers' = PhoneNumbers                          │
│ UserPhoto' = UserPhoto                                │
│ Valid' = Valid                                        │
│ Invalid' = Invalid                                    │
│ Postings' = Posting                                   │
│ HostPreferences' = HostPreferences                    │
└───────────────────────────────────────────────────────┘
```

A person can create either a user account, or they can create an admin account. The idea is that the admin account isn't something that you can specify, but you must be given a link to sign up for. The link you follow gives the admin flag.

$$CreateAccount == CreateUserAccount \lor CreateAdminAccount$$

To login, a user needs to enter their email and password. At this point they are taken to the main menu.

```
┌─ Login ──────────────────────────────────────────────┐
│ ΞHomestayDatabase                                     │
│ E? : Email                                            │
│ P? : Password                                         │
│ FN : FirstName                                        │
│ LN : LastName                                         │
│ AD : AdminFlag                                        │
│ Resp! : Response                                      │
├───────────────────────────────────────────────────────┤
│ E? ∈ Emails                                           │
│ P? ∈ Passwords                                        │
│ (E?, P?, FN, LN, AD) ∈ Applicants                    │
│ Resp! = LoginSuccessful                               │
└───────────────────────────────────────────────────────┘
```

Users can reset their password if they forget it. We take an email address,

generate a password token, then dish that off to the email address. The user then finds the email with the reset token/link and proceeds to reset their password.

This is supposed to be one more step of indirection so that the user wont have their password reset at random. Of course, if the email address is already compromised, it doesn't make much difference.

$$
\begin{array}{l}
\hline
\quad ForgotPassword \underline{\hspace{6cm}} \\
\hline
\quad \Delta HomestayDatabase \\
\quad E? : Email \\
\quad T! : PToken \\
\quad ID : \mathbb{N} \\
\quad P : Password \\
\quad FN : FirstName \\
\quad LN : LastName \\
\quad AD : AdminFlag \\
\hline
\quad E? \in Emails \\
\quad (E?, P, FN, LN, AD) \in Applicants \\
\quad ID = \#\, Valid + \#\, Invalid + 1 \\
\quad T! = ID \mapsto E? \\
\quad Valid' = Valid \cup \{T!\} \\
\quad Invalid' = Invalid \\
\quad Applicants' = Applicants \\
\quad Emails' = Emails \\
\quad Passwords' = Passwords \\
\quad PhoneNumbers' = PhoneNumbers \\
\quad UserPhoto' = UserPhoto \\
\quad Postings' = Postings \\
\quad HostPreferences' = HostPreferences \\
\hline
\end{array}
$$

Once the user has the password token, they can enter their new password. We let them know that the reset was successful.

```
┌─ ResetPassword ──────────────────────────────────
│ ΔHomestayDatabase
│ P?, P : Password
│ T? : PToken
│ Resp! : Response
│ ID : ℕ
│ E : Email
│ FN : FirstName
│ LN : LastName
│ AD : AdminFlag
│ App₀, App₁ : Applicant
├──────────────────────────────────────────────────
│ T? ∈ Valid
│ E = second  T?
│ Valid' = Valid \ { T? }
│ Invalid' = Invalid ∪ { T? }
│ App₀ = (E, P, FN, LN, AD) ∈ Applicants
│ App₁ = (E, P?, FN, LN, AD)
│ Applicants' = Applicants ⊗ App₁
│ Resp! = PasswordResetSuccessful
│ Emails' = Emails
│ Passwords' = Passwords
│ PhoneNumbers' = PhoneNumbers
│ Postings' = Postings
│ HostPreferences' = HostPreferences
└──────────────────────────────────────────────────
```

3   Student

section *student* **parents** *general*

4 Host

The host section allows the user to view and modify preferences specific to hosting a location. There are some general preferences for each host, e.g. smoking, and pets. Then there are preferences for each host location, e.g. price and availability.

4.1 Preferences

$$updateHostPrefs : \mathbb{P}\, HostPreference \times HostPreference \to \mathbb{P}\, HostPreference$$

$$\forall\, hp : HostPreference;\ hps : \mathbb{P}\, HostPreference \bullet$$
$$\exists\, hp_0 : HostPreference \mid$$
$$firstOf6\ hp_0 = firstOf6\ hp \bullet$$
$$updateHostPrefs(hps, hp) = (hps \setminus \{hp_0\}) \cup \{hp\}$$

__ *EditHostSmoking* _____

$\Delta HomestayDatabase$
$H? : HostPreference$
$H : HostPreference$
$S? : Smoking$

$H? \in HostPreferences$
$H = updateSecondOf6(H?, S?)$
$HostPreferences' = updateHostPrefs(HostPreferences, H)$
$Applicants' = Applicants$
$Emails' = Emails$
$Passwords' = Passwords$
$PhoneNumbers' = PhoneNumbers$
$UserPhotos' = UserPhotos$
$Valid' = Valid$
$Invalid' = Invalid$
$Postings' = Postings$

```
┌─ EditHostPets ────────────────────────────────────┐
│ ΔHomestayDatabase                                  │
│ H?, H : HostPreference                             │
│ P? : Pets                                          │
├────────────────                                    │
│ H? ∈ HostPreferences                               │
│ H = updateThirdOf6(H?, P?)                         │
│ HostPreferences' = updateHostPrefs(HostPreferences, H) │
│ Applicants' = Applicants                           │
│ Emails' = Emails                                   │
│ Passwords' = Passwords                             │
│ PhoneNumbers' = PhoneNumbers                       │
│ UserPhotos' = UserPhotos                           │
│ Valid' = Valid                                     │
│ Invalid' = Invalid                                 │
│ Postings' = Postings                               │
└────────────────────────────────────────────────────┘
```

```
┌─ EditHostChildren ────────────────────────────────┐
│ ΔHomestayDatabase                                  │
│ H?, H : HostPreference                             │
│ C? : Children                                      │
├────────────────                                    │
│ H? ∈ HostPreferences                               │
│ H = updateFourthOf6(H?, C?)                        │
│ HostPreferences' = updateHostPrefs(HostPreferences, H) │
│ Applicants' = Applicants                           │
│ Emails' = Emails                                   │
│ Passwords' = Passwords                             │
│ PhoneNumbers' = PhoneNumbers                       │
│ UserPhotos' = UserPhotos                           │
│ Valid' = Valid                                     │
│ Invalid' = Invalid                                 │
│ Postings' = Postings                               │
└────────────────────────────────────────────────────┘
```

```
┌─ EditHostDiet ──────────────────────────────────
│ ΔHomestayDatabase
│ H?, H : HostPreference
│ D? : Diet
├─────────────
│ H? ∈ HostPreferences
│ H = updateFifthOf6(H?, D?)
│ HostPreferences' = updateHostPrefs(HostPreferences, H)
│ Applicants' = Applicants
│ Emails' = Emails
│ Passwords' = Passwords
│ PhoneNumbers' = PhoneNumbers
│ UserPhotos' = UserPhotos
│ Valid' = Valid
│ Invalid' = Invalid
│ Postings' = Postings
└──────────────────────────────────────────────────
```

```
┌─ EditHostReligion ──────────────────────────────
│ ΔHomestayDatabase
│ H?, H : HostPreference
│ R? : Religion
├─────────────
│ H? ∈ HostPreferences
│ H = updateSixthOf6(H?, R?)
│ HostPreferences' = updateHostPrefs(HostPreferences, H)
│ Applicants' = Applicants
│ Emails' = Emails
│ Passwords' = Passwords
│ PhoneNumbers' = PhoneNumbers
│ UserPhotos' = UserPhotos
│ Valid' = Valid
│ Invalid' = Invalid
│ Postings' = Postings
└──────────────────────────────────────────────────
```

$EditHostPreferences == EditHostSmoking \lor$
$\quad EditHostPets \lor$
$\quad EditHostChildren \lor$
$\quad EditHostDiet \lor$
$\quad EditHostReligion$

4.2 Validation

So long as the start date is before the end date, we consider the Availability to
be valid.

```
ValiDateGood
Avail? : Availability
AvailResp! : Response
Start, End : Date
Y_0, Y_1 : Year
M_0, M_1 : Month
D_0, D_1 : Day
```
$((M_0, D_0, Y_0), (M_1, D_1, Y_1)) = Avail?$
$(Y_0 < Y_1) \lor$
$\quad (Y_0 = Y_1 \land month2Nat\ M_0 < month2Nat\ M_1) \lor$
$\quad (Y_0 = Y_1 \land M_0 = M_1 \land D_0 < D_1)$
$AvailResp! = ValidAvailability$

If the start date is after the end date, then no bueno.

```
ValiDateBad
Avail? : Availability
AvailResp! : Response
Start, End : Date
Y_0, Y_1 : Year
M_0, M_1 : Month
D_0, D_1 : Day
```
$((M_0, D_0, Y_0), (M_1, D_1, Y_1)) = Avail?$
$(Y_0 > Y_1) \lor$
$\quad (Y_0 = Y_1 \land month2Nat\ M_0 > month2Nat\ M_1) \lor$
$\quad (Y_0 = Y_1 \land M_0 = M_1 \land D_0 > D_1)$
$AvailResp! = InvalidAvailability$

$ValiDate == ValiDateGood \lor ValiDateBad$

4.3   Postings

To create a new posting, the host enters all the relevant information for hosting

```
┌─ NewPosting ─────────────────────────────────────
│ ΔHomestayDatabase
│ ValiDate
│ H? : HostPreference
│ Addr? : Address
│ E? : ExactOrArea
│ WIDTH? : ℕ
│ LENGTH? : ℕ
│ PR? : Price
│ Avail? : Availability
│ PH? : Photo
│ P : Posting
│ AI : AddressInfo
│ S : SizeOfRoom
│ AvailResp! : Response
├──────────────────────────────────────────────────
│ AvailResp! = ValidAvailability
│ AI = Addr? ↦ E?
│ S = WIDTH? ↦ LENGTH?
│ P = (H?, AI, S, PR?, Avail?, PH?)
│ Postings' = Postings ∪ {P}
│ Applicants' = Applicants
│ Emails' = Emails
│ Passwords' = Passwords
│ PhoneNumbers' = PhoneNumbers
│ UserPhotos' = UserPhotos
│ Valid' = Valid
│ Invalid' = Invalid
│ HostPreferences' = HostPreferences
└──────────────────────────────────────────────────
```

Deleting a post removes the post from Postings

```
┌─ DeletePosting ──────────────────────────────────────────
│ ΔHomestayDatabase
│ P? : Posting
├──────────────
│ P? ∈ Postings
│ Postings' = Postings \ {P?}
│ Applicants' = Applicants
│ Emails' = Emails
│ Passwords' = Passwords
│ PhoneNumbers' = PhoneNumbers
│ UserPhotos' = UserPhotos
│ Valid' = Valid
│ Invalid' = Invalid
│ HostPreferences' = HostPreferences
└──────────────────────────────────────────────────────────
```

5   Administrator

section *admin* **parents** *general*

Administrator login only needs to act like an inbox with "flag as inappropriate" or "accept/reject match" messages from students. The administrator will have a link to the group chat that is flagged and can add messages or end the chat if it is inappropriate. If the student and host decide to chat, once they send in their accept/reject match to the admin, the admin will have final approval to accept or reject the match.

$$AcceptMessage ::=$$
$$OneAccept$$
$$|\ TwoAccepts$$
$$|\ NoAccepts$$
$$|\ Inappropriate$$

$AdminMatch : AcceptMessage \nrightarrow MatchFlag$

$AdminMatch\ OneAccept = Reject\ \wedge$
$AdminMatch\ NoAccepts = Reject\ \wedge$
$AdminMatch\ TwoAccepts = Accept$

When we need to get rid of a user, we have some steps to take

- Remove the applicant.

- Remove the email corresponding to the applicant.

- Remove the password corresponding to the applicant.

```
┌─ RemoveUser ──────────────────────────────────────
│ ΔHomestayDatabase
│ A? : Applicant
│ E : Email
│ P : Password
├──────────────────────────────────────────────────
│ E = firstOf5 A?
│ P = secondOf5 A?
│ Applicants' = Applicants \ {A?}
│ Emails' = Emails \ {E}
│ Passwords' = Passwords \ {P}
│ HostPreferences' = HostPreferences
│ PhoneNumbers' = PhoneNumbers
│ UserPhotos' = UserPhotos
│ Valid' = Valid
│ Invalid' = Invalid
│ Postings' = Postings
└──────────────────────────────────────────────────
```

```
┌─ AdminInappropriate ──────────────────────────────
│ ΔHomestayDatabase
│ M? : AcceptMessage
│ E? : Email
│ I : IFlag
│ In, In2 : InappropriateFlag
├──────────────────────────────────────────────────
│ M? = Inappropriate
│ {E? ↦ In} = {E?} ◁ IFlags
│ In ≠ Remove
│ In2 = Nat2InFlag((InFlag2Nat In) + 1)
│ IFlags' = IFlags ⊕ {E? ↦ In2}
│ In2 ≠ Remove
│ Applicants' = Applicants
│ Emails' = Emails
│ Passwords' = Passwords
│ PhoneNumbers' = PhoneNumbers
│ UserPhotos' = UserPhotos
│ Valid' = Valid
│ Invalid' = Invalid
└──────────────────────────────────────────────────
```

$\underline{\quad InappropriateUser \quad}$
$\Delta HomestayDatabase$
$RemoveUser$
$M? : AcceptMessage$
$E? : Email$
$I : IFlag$
$In, In2 : InappropriateFlag$

$M? = Inappropriate$
$\{E? \mapsto In\} = \{E?\} \lhd IFlags$
$In \neq Remove$
$In2 = Nat2InFlag((InFlag2Nat\ In) + 1)$
$IFlags' = IFlags \oplus \{E? \mapsto In2\}$
$In2 = Remove$

6 Group Chat

We want the user to be able to see all the chats they are a part of. When they see the chats, they should have a few options.

- Add a new message to a chat.

- Approve the union of Student and Host synergy.

- Disapprove the union of Student and Host synergy.

- Delete the chat.

$AddMessage$ _____
$\Delta HomestayDatabase$
$From?, To? : Applicant$
$Msg? : Text$

_____

$Chats' = Chats \cup \{(From?, To?) \mapsto Msg?\}$
$Applicants' = Applicants$
$Emails' = Emails$
$HostPreferences' = HostPreferences$
$IFlags' = IFlags$
$Invalid' = Invalid$
$Matches' = Matches$
$Passwords' = Passwords$
$PhoneNumbers' = PhoneNumbers$
$Postings' = Postings$
$UserPhotos' = UserPhotos$
$Valid' = Valid$

```
┌─ ApproveSynergy ──────────────────────────────────────────
│ ΔHomestayDatabase
│ Match? : MatchFlag
│ App₀?, App₁? : Applicant
├───────────
│ Match? = Accept
│ Matches' = Matches ∪ {(App₀?, App₁?, Match?)}
│ Applicants' = Applicants
│ Chats' = Chats
│ Emails' = Emails
│ HostPreferences' = HostPreferences
│ IFlags' = IFlags
│ Invalid' = Invalid
│ Passwords' = Passwords
│ PhoneNumbers' = PhoneNumbers
│ Postings' = Postings
│ UserPhotos' = UserPhotos
│ Valid' = Valid
└───────────────────────────────────────────────────────────
```

```
┌─ DisapproveSynergy ───────────────────────────────────────
│ ΔHomestayDatabase
│ Match? : MatchFlag
│ App₀?, App₁? : Applicant
├───────────
│ Match? = Reject
│ Matches' = Matches ∪ {(App₀?, App₁?, Match?)}
│ Applicants' = Applicants
│ Chats' = Chats
│ Emails' = Emails
│ HostPreferences' = HostPreferences
│ IFlags' = IFlags
│ Invalid' = Invalid
│ Passwords' = Passwords
│ PhoneNumbers' = PhoneNumbers
│ Postings' = Postings
│ UserPhotos' = UserPhotos
│ Valid' = Valid
└───────────────────────────────────────────────────────────
```

```
┌─ DeleteChat ──────────────────────────────────────────
│ ΔHomestayDatabase
│ From?, To? : Applicant
├───────────────────────────────────────────────────────
│ Chats' = {(From?, To?)} ⩤ Chats
│ Applicants' = Applicants
│ Emails' = Emails
│ HostPreferences' = HostPreferences
│ IFlags' = IFlags
│ Invalid' = Invalid
│ Matches' = Matches
│ Passwords' = Passwords
│ PhoneNumbers' = PhoneNumbers
│ Postings' = Postings
│ UserPhotos' = UserPhotos
│ Valid' = Valid
└───────────────────────────────────────────────────────
```

7 Profile

The profile section has functionality to update/change basic user information: First Name, Last Name, User Photo, Email, and Phone Number.

We just need to replace the first name for the applicant.

―― *EditFirstName* ―――――――――――――――――――――――――
$\Delta HomestayDatabase$
$FN? : FirstName$
$App_0?, App_1 : Applicant$
――――――――
$App_0? \in Applicants$
$App_1 = updateThirdOf5(App_0?, FN?)$
$Applicants' = Applicants \otimes App_1$
$Emails' = Emails$
$Passwords' = Passwords$
$PhoneNumbers' = PhoneNumbers$
$Valid' = Valid$
$Invalid' = Invalid$
$HostPreferences' = HostPreferences$
$Postings' = Postings$
$UserPhotos' = UserPhotos$
―――――――――――――――――――――――――――――――――――

We just need to replace the last name for the applicant.

―― *EditLastName* ―――――――――――――――――――――――――
$\Delta HomestayDatabase$
$LN? : LastName$
$App_0?, App_1 : Applicant$
――――――――
$App_0? \in Applicants$
$App_1 = updateFourthOf5(App_0?, LN?)$
$Applicants' = Applicants \otimes App_1$
$Emails' = Emails$
$Passwords' = Passwords$
$PhoneNumbers' = PhoneNumbers$
$UserPhotos' = UserPhotos$
$Valid' = Valid$
$Invalid' = Invalid$
$HostPreferences' = HostPreferences$
$Postings' = Postings$
―――――――――――――――――――――――――――――――――――

We just need to replace the photo for the applicant.

```
┌─ EditPhoto ──────────────────────────────────────────────
│ ΔHomestayDatabase
│ UP? : UserPhoto
│ App? : Applicant
├──────────────────────────────────────────────────────────
│ UserPhotos′ = UserPhotos ⊕ {App? ↦ UP?}
│ Applicants′ = Applicants
│ Emails′ = Emails
│ Passwords′ = Passwords
│ PhoneNumbers′ = PhoneNumbers
│ Valid′ = Valid
│ Invalid′ = Invalid
│ HostPreferences′ = HostPreferences
│ Postings′ = Postings
└──────────────────────────────────────────────────────────
```

We have to check that the email is not currently used in the system. Then update the email for the applicant and ensure the changes cascade throughout the system.

```
┌─ EditEmail ──────────────────────────────────────────────
│ ΔHomestayDatabase
│ FN : FirstName
│ LN : LastName
│ P : Password
│ E?, E : Email
│ AD : AdminFlag
├──────────────────────────────────────────────────────────
│ ∃ App_0, App_1 : Applicant | App_0 = (E, P, FN, LN, AD) ∈ Applicants ∧
│ App_1 = (E?, P, FN, LN, AD) ∉ Applicants •
│ Applicants′ = Applicants \ {App_0} ∧ Applicants′ = Applicants ∪ {App_1}
│ Emails′ = Emails
│ Passwords′ = Passwords
│ PhoneNumbers′ = PhoneNumbers
│ UserPhotos′ = UserPhotos
│ Valid′ = Valid
│ Invalid′ = Invalid
│ HostPreferences′ = HostPreferences
│ Postings′ = Postings
└──────────────────────────────────────────────────────────
```

We just need to update the phone number for the applicant.

```
┌─ EditPhone ──────────────────────────────────────────────────┐
│ ΔHomestayDatabase                                             │
│ PNUM? : Phone                                                 │
│ App? : Applicant                                              │
├───────────────────────────────────────────────────────────── │
│ PhoneNumbers' = PhoneNumbers ⊕ {App? ↦ PNUM?}                 │
│ Applicants' = Applicants                                      │
│ Emails' = Emails                                              │
│ Passwords' = Passwords                                        │
│ UserPhotos' = UserPhotos                                      │
│ Valid' = Valid                                                │
│ Invalid' = Invalid                                            │
│ HostPreferences' = HostPreferences                           │
│ Postings' = Postings                                          │
└──────────────────────────────────────────────────────────────┘
```