

"Bonjour" Product Test Plan

Test Plan

Passing all "Critical" test cases will give a working product, passing all "Critical" and "Important" test cases will give a fully functional product, and passing all test cases will make the product strongest to the best of our knowledge. However, the server component must force all the test cases to be checked, regardless of its level of severity.

User Identity

"User identity" is the set of functionalities the user identifies oneself with the application. The basic behavior and properties include:

Properties:

- i. User email is a string of length 3 to 255 of the form
[-0-9a-zA-Z.+_]+@[-0-9a-zA-Z. _]+\.[a-zA-Z]{2,}
- ii. User password is a string of length 6 to 32, allowing special characters.
- iii. User nickname is a string of length 4 to 20 of the form
[0-9a-zA-Z. _]{4, 20}
- iv. User age must be greater than 13.

Behavior:

1. User registration (needs properties i., ii., and iv)
2. User log in
3. Password reset
4. Profile update

User.R01			Not started
Classification	Functionality	Severity	Critical
Input / Instructions	Click the register button on first screen of the app		
Expected Outcome	The register activity shows up		
Comment	Critical at first test, workaround afterwards.		

User.R02			Not started
Classification	Functionality & Equivalence	Severity	Critical
Input / Instructions	<ol style="list-style-type: none"> 1. Type a valid user email. E.g., "bu1@purdue.edu" 2. Type a password. E.g., "HailPurdue!" 3. Type a valid age. E.g., 20 4. Check the "I agree the terms of using this app" box. 5. Click "Register" button 		

Expected Outcome	When the "Register" button is clicked, the fields are encrypted with RSA algorithm and sent to the server, and the server can decrypt the information to the original input.
Comment	

User. R03			Not started
Classification	Functionality & Equivalence	Severity	Critical
Input / Instructions	1. Type all the invalid fields listed below on the registration form, one per time		
Expected Outcome	1. All the invalid formats should be reported by the app. 2. The server must refuse the registration request whenever there is at least one invalid field.		
Comment	Use User.properties.email.invalid*, User.properties.pwdLen*, and User.properties.minAge* to run this test case.		

User.properties.email.valid			Not started
Classification	Equivalence	Severity	Critical
Input / Instructions	1. Test the following email addresses <u>one by one</u> : <ul style="list-style-type: none"> a@b.c 1a@2b.com A1.b2@c3.cn A1.B2+c3_d4@e5_f6.com.cn 		
Expected Outcome	The email addresses above should be reported valid.		
Comment	This set of strings covers the class of valid format.		

User.properties.email.invalid1			Not started
Classification	Equivalence & Boundary Values	Severity	Workaround
Input / Instructions	1. Test the following strings of invalid format <u>one by one</u> : <ul style="list-style-type: none"> a@b.c. (invalid format, missing root server) a@.b (invalid format, missing the domain name) @a.b (invalid format, missing username) a.b.c (invalid format, missing '@' character) ab (invalid format, not an email address) 		
Expected Outcome	The email addresses above should be reported invalid and the app should ask for retyping.		
Comment	This set of strings covered the class of missing parts of the email address as defined in RFC5322.		

User.properties.email.invalid2			Not started
Classification	Equivalence	Severity	Workaround
Input / Instructions	1. Test the following strings of invalid format <u>one by one</u> : <ul style="list-style-type: none"> • a @b.c (invalid format, containing space char ' ') • a@ .c (invalid format, as above) • a@b. (invalid format, as above) • a>@<b.= (invalid format, other invalid characters) • クラウド@gmail.com (invalid format, non-ASCII chars) 		
Expected Outcome	The email addresses above should be reported invalid and the app should ask for retyping.		
Comment	This set of strings covered the class of having disallowed chars.		

User.properties.email.invalid3			Not started
Classification	Boundary Values	Severity	Workaround
Input / Instructions	1. Test the following strings of different lengths <u>one by one</u> : <ul style="list-style-type: none"> • Note that the lower bound has been tested in User.r03.email.invalid1 • UserName10UserName10UserName10UserName10 UserName10UserName10UserName10UserName10 UserName10UserName10UserName10UserName10@ UserName10UserName10UserName10UserName10 UserName10UserName10UserName10UserName10 UserName10UserName10UserName10UserName10 2345678901234.6 (length of 256 exceeds the limit) 		
Expected Outcome	The email addresses above should be reported invalid and the app should ask for retyping.		
Comment	This test case covers testing the upper bound length and lower bound length of an email field.		

User.properties.pwdLen.1			Not started
Classification	Equivalence & Boundary Value	Severity	Critical
Input / Instructions	1. Test the following strings for password field <u>one by one</u> : <ul style="list-style-type: none"> • abcdef (length = 6) • abcdefg (length = 7) • abcde fghij abcde+fghij-1234567 (length = 31) • abcde fghij abcde+fghij-12345678 (length = 32) 		
Expected Outcome	The strings should be reported valid as passwords.		
Comment	Equivalence class of length [6,32] with boundary values		

User.properties.pwdLen.2			Not started
Classification	Equivalence & Boundary Value	Severity	Important
Input / Instructions	1. Test the following strings for password field <u>one by one</u> : <ul style="list-style-type: none"> "" (length = 0) Aaaaa (length = 5) 		
Expected Outcome	The strings should be reported invalid as passwords, and the app should ask the user to retype.		
Comment	Equivalence class of length [0, 5] with boundary values		

User.properties.pwdLen.3			Not started
Classification	Equivalence & Boundary Value	Severity	Important
Input / Instructions	1. Test the following strings for password field <u>one by one</u> : <ul style="list-style-type: none"> abcde fghij abcde+fghij-123456789 (length = 33) 		
Expected Outcome	The string should be reported invalid as password, and the app should ask the user to retype.		
Comment	Equivalence class of length [33, ∞] with boundary value 33		

User.properties.ageMin.1			Not started
Classification	Equivalence & Boundary Value	Severity	Critical
Input / Instructions	1. Test the following numbers in the age field <u>one by one</u> : <ul style="list-style-type: none"> 14 15 		
Expected Outcome	The numbers should be valid as user age.		
Comment	Equivalence class of age {x x ≥ 14, x int} with boundary value 14		

User.properties.ageMin.2			Not started
Classification	Equivalence & Boundary Value	Severity	Important
Input / Instructions	1. Test the following numbers in the age field <u>one by one</u> : <ul style="list-style-type: none"> -1 0 13 		
Expected Outcome	The numbers should not be accepted as age values. The app should prompt for retyping.		
Comment	Equivalence class of int {x x < 13, x int}		

User.properties.ageMin.3			Not started
Classification	Equivalence	Severity	Workaround
Input / Instructions	1. Test the following number in the age field: <ul style="list-style-type: none"> 13.9 		
Expected Outcome	The numbers should not be accepted as age values because it is		

	not an int value. The app should prompt for retyping.
Comment	Equivalence class of non-integer numbers

User.l01			Not started
Classification	Functionality & Equivalence	Severity	Critical
Input / Instructions	<ol style="list-style-type: none"> 1. Enter the correct user email and corresponding password 2. Click the "Log in" button 		
Expected Outcome	<ol style="list-style-type: none"> 1. The app first checks if the email and password contain valid data; 2. The app encrypts the information using RSA algorithm and send it to the server; 3. The server can correctly decrypt the information to their original text; 4. The server can correctly compare the information with the record in database, and find the pair provided; 5. The server can correctly return a success signal with the user's profile to the app; 6. The app can correctly parse the signal and user profile, log user in, and enters into the user's home page. 		
Comment	Each step of the outcome must be checked.		

User.l02.invalidData1			Not started
Classification	Equivalence	Severity	Workaround
Input / Instructions	<ol style="list-style-type: none"> 1. Enter an email of invalid format and a valid password 2. Click the "Log in" button 		
Expected Outcome	<ol style="list-style-type: none"> 1. The app checks the format of the fields and report them as invalid; 2. Ask the user to retype. 		
Comment	Use User.properties.email.invalid* test cases to test the email.		

User.l02.invalidData			Not started
Classification	Equivalence	Severity	Workaround
Input / Instructions	<ol style="list-style-type: none"> 1. Enter a valid email but a password of invalid format 2. Click the "Log in" button 		
Expected Outcome	<ol style="list-style-type: none"> 3. The app checks the format of the fields and report them as invalid; 		

	4. Ask the user to retype.
Comment	Use User.properties.pwdLen.* test cases to test the password.

User.I03			Not started
Classification	Functionality & Equivalence	Severity	Critical
Input / Instructions	1. Enter a valid email and a valid password, but not registered 2. Click the "Log in" Button		
Expected Outcome	1. The server reports the email address as "Either not registered or wrong password". 2. The app pops up a message telling log in failed.		
Comment	Combine the report as "Either not registered or wrong password" to avoid revealing if an email is in the user pool.		

User.I04			Not started
Classification	Functionality & Equivalence	Severity	Critical
Input / Instructions	1. Enter a valid email but a valid, wrong password. 2. Click the "Log in" Button		
Expected Outcome	1. The server reports the pair as "Either not registered or wrong password". 2. The app pops up a message telling log in failed.		
Comment			

User.rp1.a			Not started
Classification	Functionality & Equivalence	Severity	Important
Input / Instructions	1. Click "Reset my password" 2. In the following screen, type a registered email address.		
Expected Outcome	1. An email with password reset link is sent to the email address. 2. A notification pops up displaying "A link has been sent to your email address".		
Comment			

User.rp1.b			Not started
Classification	Functionality & Equivalence	Severity	Workaround
Input / Instructions	1. Click "Reset my password" 2. In the following screen, type a valid email address that is not registered.		

Expected Outcome	<ol style="list-style-type: none"> 1. The server realizes the fact that the email is not in user database, and does nothing. 2. The app pops up a message displaying: "The email address is not registered".
Comment	Use same app response as in User.rp1.a to protect user privacy.

User.rp1.c			Not started
Classification	Functionality & Equivalence	Severity	Workaround
Input / Instructions	<ol style="list-style-type: none"> 1. Click "Reset my password" 2. In the following screen, type an invalid email address. 		
Expected Outcome	<ol style="list-style-type: none"> 1. The app reports the string is not a valid email address, and asks for retyping. 		
Comment	Use User.properties.email.invalid* test cases to test the email.		

User.ProfileUpdate01.Change			Not Started
Classification	Functionality & Equivalence	Severity	2. Important
Input / Instructions	<ol style="list-style-type: none"> 1. Click "My Profile" button 2. Click on edit profile button to add new information or change existing information. 3. At the end the user can click on "Confirm Changes" 4. A small window pops up with the options "Yes" and "No". 5. If user chooses "Yes" then the changes are committed and the user can view the updates profile page. But if the user chooses "No" then he is directed back to the edit profile page where can once again edit the information. Once they are satisfied then they can click on "Confirm Changes". 		
Expected Outcome	<ol style="list-style-type: none"> 1.If the user clicks on "Yes", the app encrypts the information using RSA algorithm and sends it to the server. 2. The server receives the request, decrypts the information to the original texts and updates the existing information for a particular user with the new information The changes are committed to the database. 3. If the user clicks on "No", the app redirects the user back to the edit profile page. 		
Comment			

User.ProfileUpdate02.ChangeLeave			Not Started
Classification	Functionality & Equivalence	Severity	1. Critical
Input / Instructions	1. Click "My Profile" button 2. Click on "Edit Profile" button to add new information or change existing information. 3. Instead of clicking on "Confirm Changes" click on "Settings" button.		
Expected Outcome	1. The app realizes that the new information has not been committed and asks user to "Confirm Changes".		
Comment			

User.ProfileUpdate03.ChangeBlank			Not Started
Classification	Equivalence & Boundary Values	Severity	1. Critical
Input / Instructions	1. Click "My Profile" button 2. Click on "Edit Profile" button to add new information or change existing information but instead of filling all blanks, leave a one or more blanks empty. 3. Click on "Confirm Changes"		
Expected Outcome	1. The app recognizes that one or more fields have been left blank. 2. Pop up window informs user 'Information Incomplete' and directs user back to "Edit Profile" page. 3. The blank field is highlighted red.		
Comment			

Matching

The matching consists of the following properties:

- UserIdentity uid {[user email, password] | [token]}
- LocationPoint loc {x, y, z, t} where
 - x, y, and z are long int or whatever GPS has to offer
 - t is of datetime type representing the time when the user is at the location (x, y, z).

The matching part consists of the following behavior:

- Notification of location gathering
- Location gathering
- Showing a list of people nearby
- View the profile of those who are close to the user

Match.Notification01.GPS			Not Started
Classification	Functionality	Severity	2. Important
Input / Instructions	1. Open "Settings" page 2. Check the checkbox for location information gathered notification when the GPS function is turned on		
Expected Outcome	1. App sends request to server. 2. Server updates the preferences in the data base 3. User gets a notification that location is being gathered and displays current location.		
Comment			

Match.Notification02.GPSInvalid			Not Started
Classification	Functionality	Severity	2. Important
Input / Instructions	1. Open "Settings" page 2. Check the checkbox for location information gathered notification when the GPS function is turned off		
Expected Outcome	1. An error message with the button move to the GPS System setting shows up and the setting will not be saved		
Comment			

Match.Notification03.TimeInterval			Not Started
Classification	Equivalence & Boundary	Severity	1. Critical
Input / Instructions	1. Click the notification interval option in the "Settings" page 2. Enter a valid minute interval (1 – 60)		
Expected Outcome	The setting are saved correctly and no warning is displayed		
Comment			

Match.Notification04.InvalidTimeInterval			Not Started
Classification	Equivalence & Boundary	Severity	2. Important
Input / Instructions	1. Click the notification interval option 2. Test the following minute interval <i>one by one</i> : -1, 0, 61		
Expected Outcome	1. An error message shows up and the setting will not be saved 2. User is asked to re-enter the values within the desired interval		
Comment			

Match.Notification05.NewMatch			Not Started
Classification	Functionality	Severity	2. Important

Input / Instructions	Walk near another user who has the same interest with the notification turns on
Expected Outcome	The notification shows up on the screen
Comment	

Match.ViewProfile01			Not Started
Classification	Functionality	Severity	2. Important
Input / Instructions	1. Click on notification which shows that there are matches 2. After the app opens up showing the matches, click on a particular user to view profile		
Expected Outcome	1. When a profile is clicked, the server extracts the information of the matched user from the database. 2. App displays the profile of the matched user		
Comment			

C/S Communication

The user does not interact directly with this layer, so the test is focused mainly on:

- Security. Data encryption and decryption. Protocol guessing. SQL Injection attack.
- Stress test.

Comm.enc			Not Started
Classification	Functionality & Equivalence	Severity	1. Critical
Input / Instructions	Whenever any sort of information is sent to the server i.e. username and password combination, profile information, the information is encrypted using RSA algorithm		
Expected Outcome	The incoming encrypted information is decrypted by the server and then stored into the database (For example: when the user edits their information) or is compared with the database (For example: when the user attempts to log into the app)		
Comment			

Comm.ServerLoad			Status
Classification	Functionality & Equivalence	Severity	1. Critical
Input / Instructions	Multiple matches found at a location		
Expected Outcome	Server has to match all of the matches and display them to the user via notification		
Comment	This will test how much load the server can handle when there are a number of matches for a number of user. The information		

	has to be sent to the correct user. The server will have to handle multiple requests and push multiple information at the same time.
--	--