Mitt Arv - Bug Analysis

Description: Regional Numbering Standards

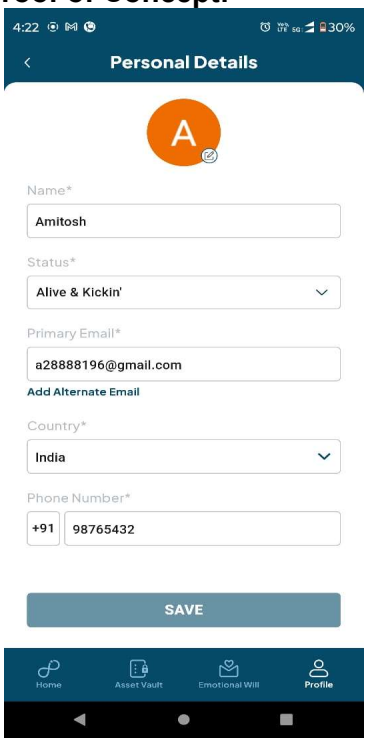
- **Steps to Reproduce:**
    1. **Regional Numbering Standards:** Some regions or countries have specific numbering standards that include 9-digit or 10 -digit mobile numbers. It's worth checking according to the country code. For example, In India We generally have 10-digit mobile number wirth country code +91. According to country code we can check whether the provided number is valid or not.

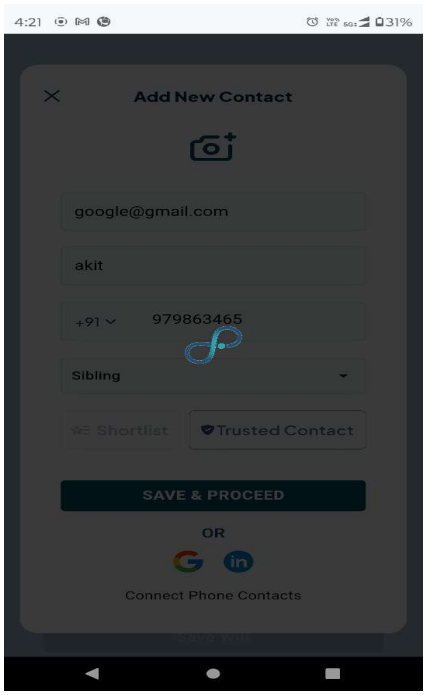
    2. **Service Provider Requirements:** The service provider you're registering with might have specific requirements for mobile numbers, such as a minimum number of digits.We should get the required data to check for this bug in the app.
- **Expected Behavior:** If someone provide 9-digit or less mobile number it should give a alert message for entering a valid mobile number]
- **Actual Behaviour:** If someone given the 9 digit or less it accepts the number.

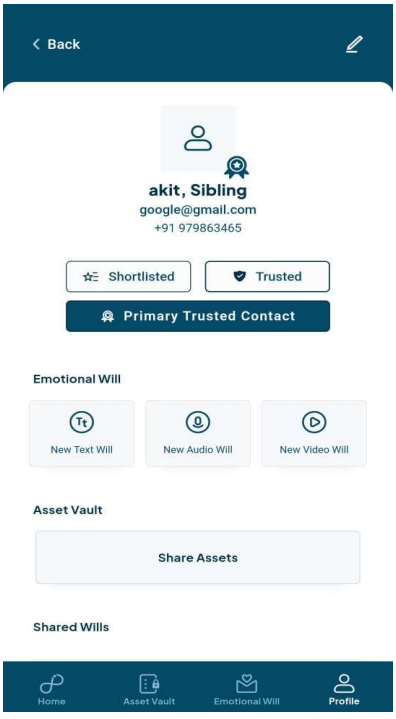
Impact

- **Potential Consequences:** If user mistakenly provide wrong data. It causes security issue like retrieving the login credential during retrieving the user profile or sensitive data, it also causes problem to contact the trusted contact in emergency situation.
- **Affected Users:** Old people are the mainly the affected users as there were unaware of these technologies or person who have less time to check for correction.
- **It also causes in continent to contact to support.**

Proof of Concept:







Better Approach:

country_code_picker: This package provides a dropdown widget to select a country and its corresponding country code. You can use the selected country code to validate the phone number using a regular expression or other validation methods.

Sample code:

```
import 'package:country_code_picker/country_code_picker.dart';

void main() {
  final phoneNumber = "12125551212";
  final selectedCountryCode = "+1"; // Assuming the selected country code is "+1"
  final usPhoneNumberRegex = r"^\d{10}$";

  final isValid = phoneNumber.startsWith(selectedCountryCode) &&
      usPhoneNumberRegex.hasMatch(phoneNumber);

  if (isValid) {
    print("Phone number is valid.");
  } else {
    print("Phone number is invalid.");
  }
}}
```

Description: Email Validation

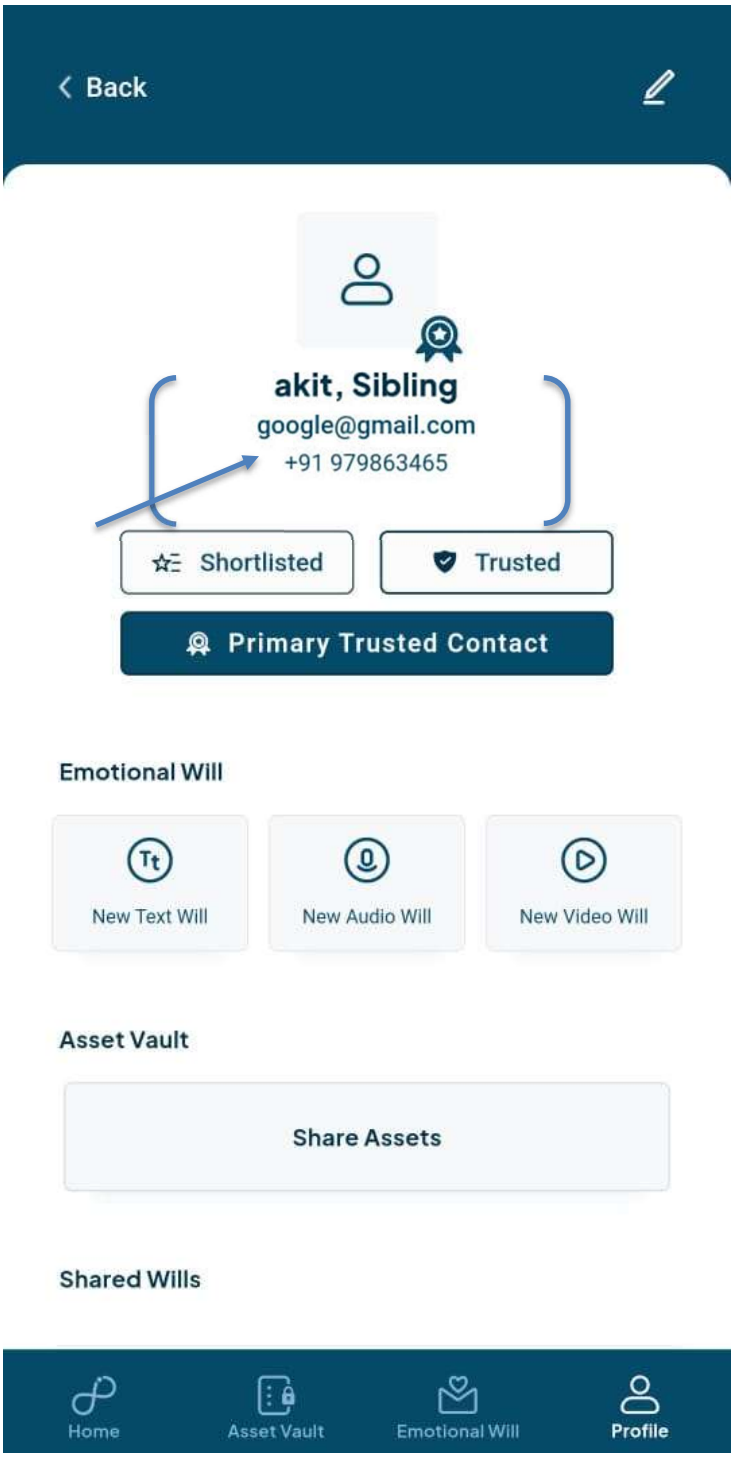
- **Steps to Reproduce:**
  1.There are several libraries available in Flutter that can be used to check for a valid email address and identify temporary or unauthorized email addresses.

- **Expected Behaviour:** If someone provide any unauthorized email or a temporary email, it should give a alert message for entering a valid mobile number]
- **Actual Behaviour:** If someone given any random or temp mail id it accepts the mail and store in contact details.

Impact

- **Potential Consequences:** If user mistakenly provide wrong data. It causes security issue like retrieving the login credential during retrieving the user profile or sensitive data, it also causes problem to contact the trusted contact in emergency situation.
- **Affected Users:** Old people are the mainly the affected users as there were unaware of these technologies or person who have less time to check for correction.

Proof of Concept:



Better Approach:

1. email_validator: This is a simple and lightweight package that provides a function to validate email addresses based on common patterns. It can be used to check if an email address is

syntactically correct.

Example:

```
import 'package:email_validator/email_validator.dart';

void main() {
  final email = "example@example.com";
  final isValid = EmailValidator.validate(email);
  print(isValid); // Output: true
}
```

2. libphonenumber-dart: This package is a Dart port of the Google libphonenumber library. It provides functions to parse, format, and validate phone numbers based on different regions. While primarily intended for phone number validation, it can also be used to check if an email address is associated with a phone number. This can be helpful in identifying temporary or disposable email addresses that might not be linked to a real person or entity.

3.

Example:

```
import 'package:libphonenumber_dart/libphonenumber_dart.dart';

void main() {
  final email = "example@example.com";
  final phoneNumberUtil = PhoneNumberUtil.getInstance();
  final phoneNumber = phoneNumberUtil.parse(email);
  final isValid = phoneNumber != null;
  print(isValid); // Output: false (assuming the email is not associated with a phone number)
}
```

3. disposable_email_validator: This package provides a database of known disposable email providers and can be used to check if an email address is associated with one of them. This can be helpful in identifying temporary or unauthorized email addresses.

Example:

```
import 'package:disposable_email_validator/disposable_email_validator.dart';

void main() {
  final email = "example@example.com";
  final isDisposable = DisposableEmailValidator.isDisposable(email);
  print(isDisposable); // Output: false (assuming the email is not a disposable email)
}
```