Software Requirements Specification

for

Purr

Karim Mohamed Omar 18102227

Mohamed Hosam Anwar 18102142

Abdelaziz Hossameldin 18100642

Fakhr El Din Mahmoud 18101187

Ahmed Abbas 18102385

Abdelrahman Emad. 18101010

A 6Niners Project

Table Of Contents

[1. Introduction 1](#_Toc61825130)

[1.1 Purpose 1](#_Toc61825131)

[1.2 Technologies Used 1](#_Toc61825132)

[1.3 Intended Audience and Reading Suggestions 1](#_Toc61825133)

[1.4 Product Scope 2](#_Toc61825134)

[1.5 References 2](#_Toc61825135)

[2. Overall Description 3](#_Toc61825136)

[2.1 Product Perspective 3](#_Toc61825137)

[2.2 Product Functions 3](#_Toc61825138)

[2.3 User Classes and Characteristics 3](#_Toc61825139)

[2.4 Operating Environment 3](#_Toc61825140)

[2.5 Design and Implementation Constraints 4](#_Toc61825141)

[2.6 User Documentation 4](#_Toc61825142)

[2.7 Assumptions and Dependencies 4](#_Toc61825143)

[3. External Interface Requirements 5](#_Toc61825144)

[3.1 System Interfaces 5](#_Toc61825145)

[3.2 Hardware Interfaces 6](#_Toc61825146)

[3.3 Software Interfaces 6](#_Toc61825147)

[3.4 Communications Interfaces 7](#_Toc61825148)

[4. System Features 7](#_Toc61825149)

[4.1 Registration 7](#_Toc61825150)

[4.1.1 Sign in: 7](#_Toc61825151)

[4.1.2 Sign up: 7](#_Toc61825152)

[4.1.3 Change password: 8](#_Toc61825153)

[4.1.4 Forgot password: 8](#_Toc61825154)

[4.1.5 Setup profile: 8](#_Toc61825155)

[4.2 Chat 9](#_Toc61825156)

[4.2.1 Send message in chat: 9](#_Toc61825157)

[4.2.2 Like a pet: 9](#_Toc61825158)

[5. Other Nonfunctional Requirements 10](#_Toc61825159)

[5.1.1 Performance Requirements: 10](#_Toc61825160)

[5.1.2 Usability: 10](#_Toc61825161)

[5.1.3 Security and Safety Requirements: 10](#_Toc61825162)

[5.1.4 Reliability and maintainability: 10](#_Toc61825163)

[5.1.5 Robustness: 10](#_Toc61825164)

[5.1.6 Business rules: 10](#_Toc61825165)

[6. Test Cases: 11](#_Toc61825166)

[7. Diagrams: 11](#_Toc61825167)

[7.1 Use Cases: 12](#_Toc61825168)

[7.2 Class Diagram: 12](#_Toc61825169)

[12](#_Toc61825170)

[7.3 Sequence Diagrams 12](#_Toc61825171)

[7.3.1 Sign up: 13](#_Toc61825172)

[7.3.2 Sign in / Logout 13](#_Toc61825173)

[7.3.3 Change Password 14](#_Toc61825174)

[7.3.4 Send Message: 14](#_Toc61825175)

# Introduction

## Purpose

This SRS’s main purpose is to create a social networking app with the soul intention of breeding and matching up users’ pets with each other.

## Technologies Used

* Dart used with Flutter for cross platform mobile development
* Python for server backend logic
* A combination of SQL and Firebase for database implementation
* Photoshop and built-in Flutter graphics for UI design

## Intended Audience and Reading Suggestions

This Document is useful to:

* Customers: To follow up with our development team if there are requirement changes.
* Developer: To ensure objectives and customer needs are fully understood.
* Software Tester: Ease of development of the required test cases for the product.
* Project Supervisor: Monitoring development progress and assistance with the teamwork when needed.

## Product Scope

**Purr** is an application made to help pets owners find potential mating partners for their pets. A cross platform design is to be implemented where versions are required to be on both the IOS App Store and Google Play Store.

When first Opening the app, new users can register and create an account. When successfully signed up, the user should be welcomed to a profile page where they fill out required categories of data for their pets, which include and are not limited to: Name, Picture, Species, Breed and Bio. There are also other non-pet related data such as: residential area, phone number, ID or Passport (to be used for security)

The app will then prompt the user a continuous stream of other registered user’s pet’s profiles with the options to either accept or deny them. If both matched users accept each other, they are redirected into a private chat room to further discuss how and where they will meet with their pets.

## References

* No references at the moment of this version’s release

**1.6 Overview**

The rest of the SRS explains the detailed specifications of **Purr**. Chapter II of the SRS presents the general factors that affect the application and its requirements, such as project constraints and user characteristics. Chapter III outlines the details of interfaces, and then the rest of the chapters' present the functional and nonfunctional requirements, Diagrams, and other related requirements.

# Overall Description

## Product Perspective

* Purr is a new self-contained standalone application, designed with a fully functional friendly user interface.
* Purr’s system utilizes firebase cloud databases for the majority of its stored data

## Product Functions

* Frequent database read and writes (chat system, new users etc..)
* User pet profile pairing with option for denial and acceptance
* Login and sign up to user pet accounts

## User Classes and Characteristics

Purr will be designed with a single user type perspective in mind, where there are no hierarchal user types in this specific system. But admin functionality will be available through a third-party system, Firebase, as most of the usable data will be stored remotely on this platform.

* The normal registered user or anyone who wishes to find a mating partner for their pet can sign up and use this application.
* Any admin will have access to Purr’s firebase account and has privileges to manage user data

## Operating Environment

Purr is designed to incorporate a cross platform design meant to function on both IOS and Android operating systems. This application is being developed through Flutter’s SDK which utilizes the Android SDK and IOS SDK for native compilation on both systems.

## Design and Implementation Constraints

* As the primary source for our data storage is through a third party company (Fire Base) if any issues were to arise in their system, it will directly affect our own. These effects include and not are not limited to, data breaches, data loss or data inaccessibility due to the inactivity of their systems

## User Documentation

No user documentation is available as of this current version release

## Assumptions and Dependencies

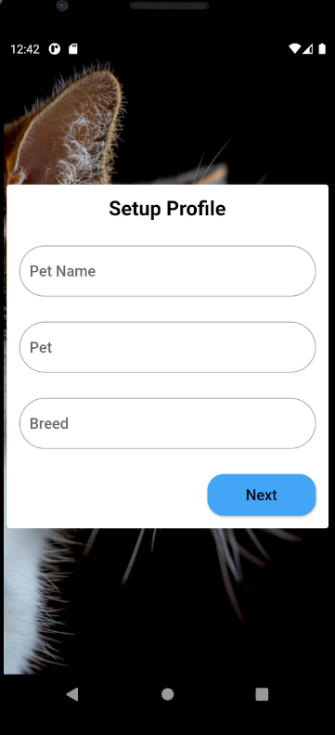
* It is assumed that all users of this application must have an active internet connection to access information from the data base
* Users will download the app from both the IOS app store and Google play store
* It is assumed that all phones have GPS capabilities when sending private locations to each other, but it is not necessary as it is consent based and will not affect the main functionality of the app

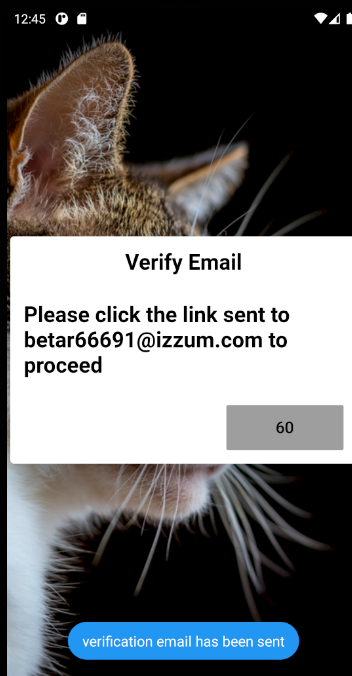
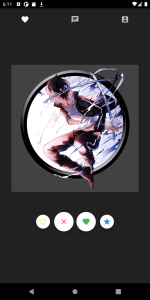
# External Interface Requirements

## System Interfaces

You are first presented with the Login and sign-up pages when opening the app for the first time. The user from there will attempt to log in to his account by typing in an email and password, otherwise a sign-up button is to be used.

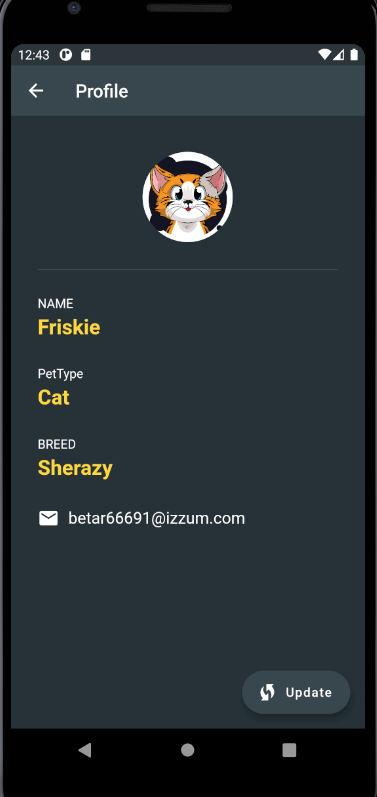
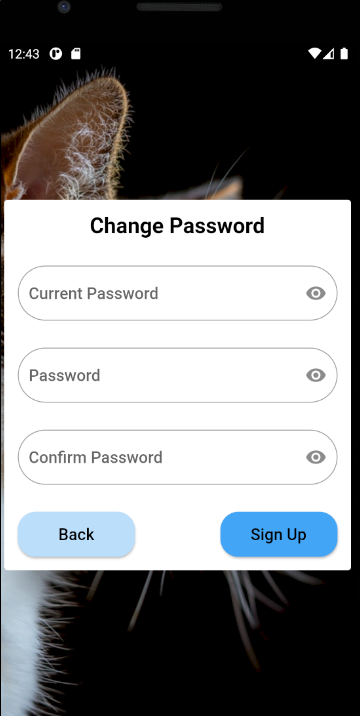
After the user has successfully logged in, they will be presented with the setup profile page from which they can add all their pet’s necessary profile information

They will then have access to the home page. This page will act as the central hub from which one the user can utilize all the features present in the application.



These many features include:

* Profile
* Change password
* Chat



## Hardware Interfaces

* A working Wi-Fi card in the devices, as Purr relies heavily on internet-based tasks
* GPS support as the user may optionally want to send or receive the other user’s location

## Software Interfaces

* Purr is a native IOS and Android application that is under development using The Flutter application development environment and the Dart programming language.
* Flutter, XCode and Android Studio packages are used for cross platform development
* CocoPods dependency manager is used to install Flutter plugins on IOS devices
* System interacts with the Firebase API to access and manage cloud data
* System uses SQLite for local data management
* System utilizes Google and Apple maps API’s for GPS services

## Communications Interfaces

* A secured HTTPS connection is to be used at all time to communicate with the Firebase data base

# System Features

## Registration

### Sign in:

* Actors: Customer
* Inputs: Customer’s Email and password
* Basic Path:
* The customer inputs his/her Email and password
* The customer clicks on sign in
* The user gets to the main app page
* Exception: the user entered a wrong email or password Display “Invalid login details”

### Sign up:

* Actors: Customer
* Inputs: Customer’s Email and password
* Basic Path:
* The customer inputs his/her Email and password and the confirmation for his password
* The customer clicks on sign up
* An email is sent to the user to confirm his email
* The user gets to the main app page
* Exception: the email Already registered Display “this email already exists”

### Change password:

* Actors: Customer
* Inputs: Customer’s old password and new password
* Pre-condition: the customer is signed in
* Basic Path:
  + The customer clicks on change pin.
  + The customer enters the old password and new password and confirmation password
  + The customer clicks on change password
  + A popup will show the user that the password got changed successfully
* Exception: the user entered a wrong old Password Display “Incorrect password”

### Forgot password:

* Actors: Customer
* Inputs: Customer’s Email
* Basic Path:
* The customer inputs his/her email
* The customer clicks on send email
* Exception: the user entered a wrong email Display “Incorrect email”

### Setup profile:

* Actors: Customer
* Inputs: Customer’s pet name, pet breed, pet type
* Pre-condition: the customer is signed in and email confirmed
* Basic Path:
  + The customer enters the pet name, pet breed, pet type
  + The customer clicks go to next page
  + The user gets to the main app page
* Exception: the user didn’t enter a field Display “cannot be left empty”

## Chat

### Send message in chat:

* Actors: Customer
* Inputs: message
* Pre-condition: the customer is signed in and email confirmed
* Basic Path:
  + The customer enters a chat he wants to send the message to
  + The customer enters the message
  + The customer clicks send
  + The message gets uploaded to firebase database
* Exception: the user didn’t enter a field Display “cannot be left empty”

### Like a pet:

* Actors: Customer
* Pre-condition: the customer is signed in and email confirmed
* Basic Path:
  + The customer clicks on like on profile
* Exception: the user didn’t enter a field Display “cannot be left empty”

# Other Nonfunctional Requirements

### Performance Requirements:

* The app needs less than 3 seconds to start.
* The app should take less than a second to operate the users’ actions.

### Usability:

* User should be able to create an account and complete profile easily in less than 2 minutes.
* User shall not take more than 10 minutes to learn how to use the app
* Communication and connection app features should be as simple as a click.

### Security and Safety Requirements:

* Store or transmit sensitive information through Google Firebase services.
* The system uses an end-to-end encryption method to secure the chat messages.
* System’s security prevents users from accessing other’s data.

### Reliability and maintainability****:****

* System in its current state is expected to have a high failure rate of 2-3 times a day.

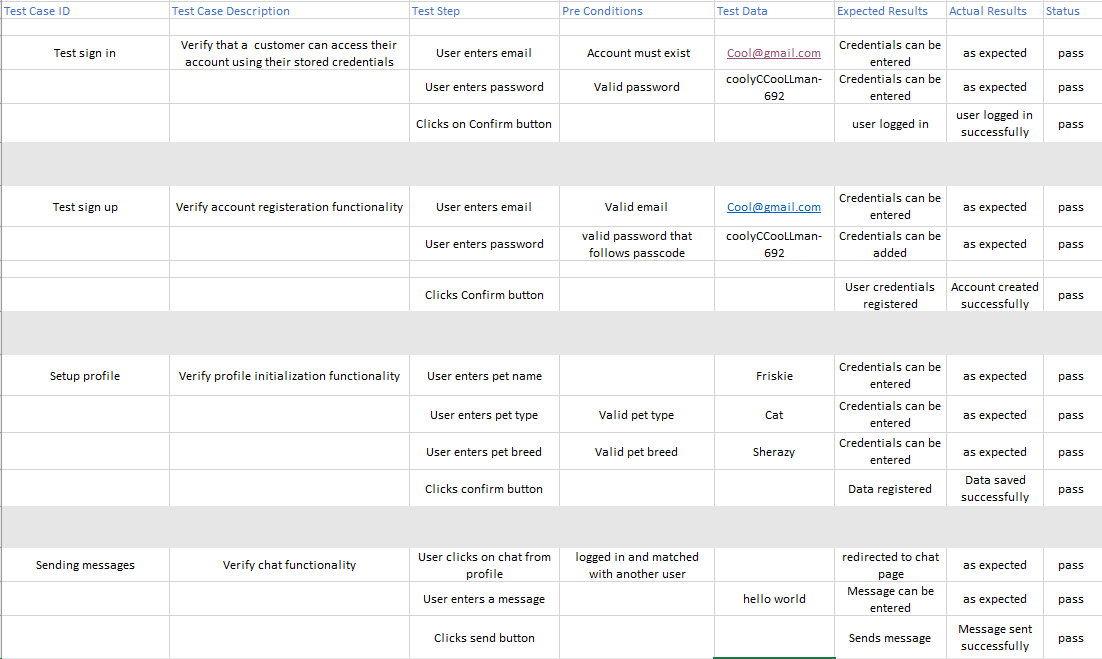
### ****Robustness:****

* System shall take 5-10 minutes to reboot in case of failure.

### ****Business rules:****

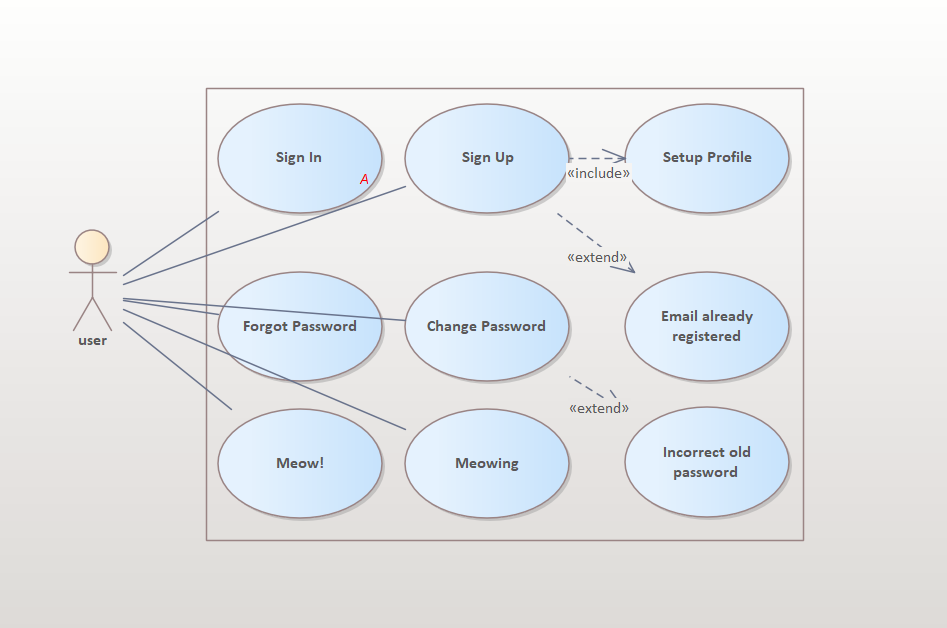
* User MUST NOT use it for profit or turn it into a breeding app, such actions will be met with harsh and legal actions. (release)

# Test Cases:



# Diagrams:

## Use Cases:

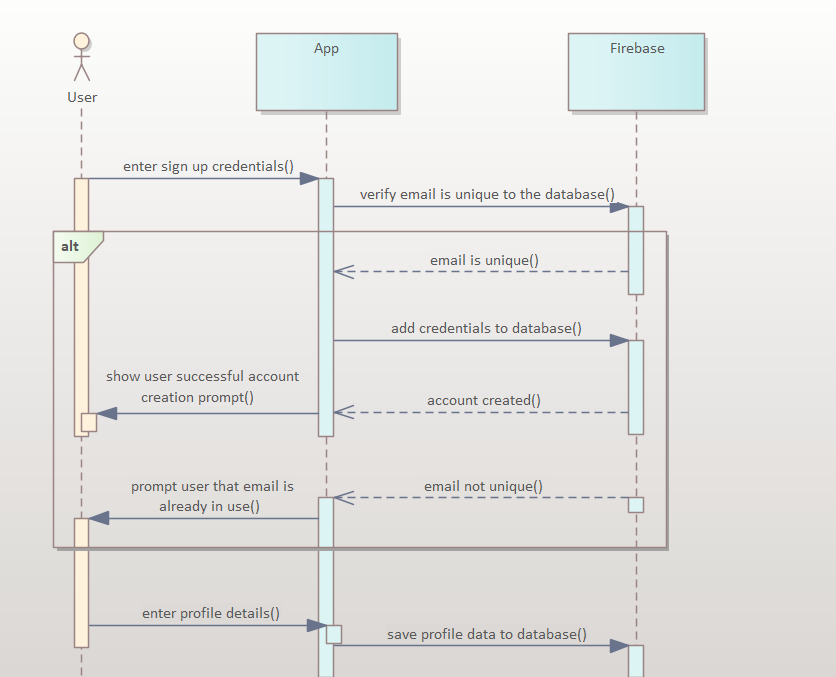


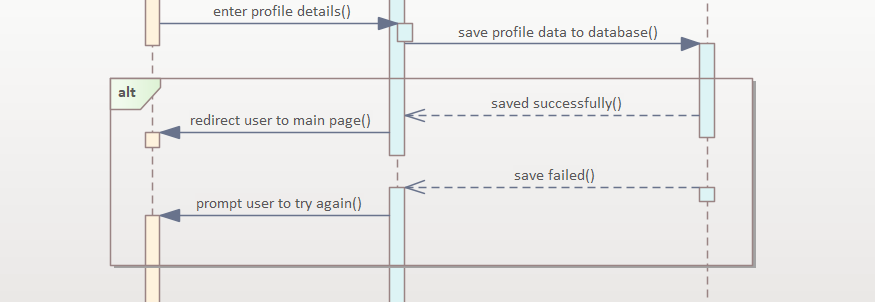
## Class Diagram:

## 

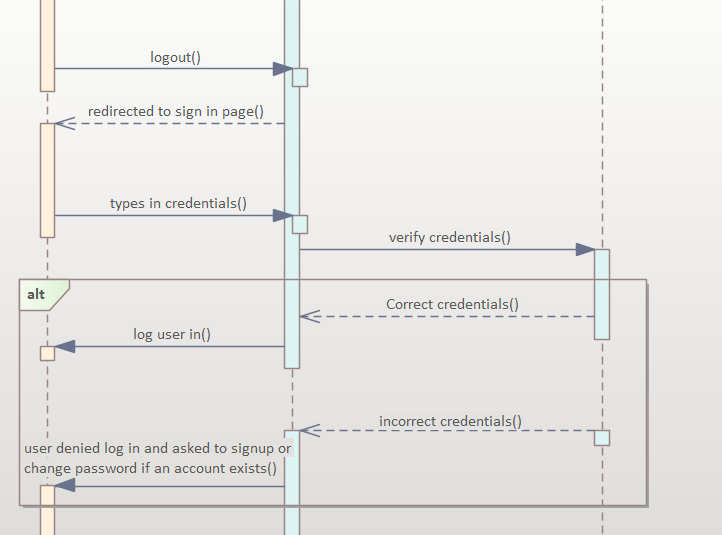
## Sequence Diagrams

### Sign up:

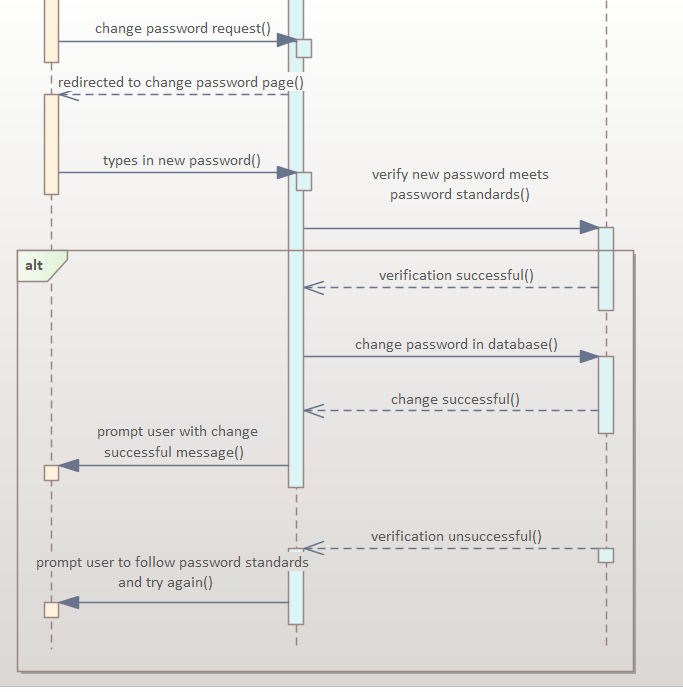




### Sign in / Logout



### Change Password



### Send Message:

