Software Design Document for S3PP version 1.0

Prepared by:

Ahmed Ehab Hamouda

Aly Moussa

Marwan Eid

Mohammed Abuelwafa

Group 7

Submitted on April 30, 2020

Copyright © 1999 by Karl E. Wiegers. Permission is granted to use, modify, and distribute this document.

Table of Contents

Revision History

Name	**Date**	**Reason for Changes**	**Version**
S3PP SDD	April 30, 2020	Initial Draft	1.0

1. Introduction

1.1 Purpose

This software design document describes the architecture and system design of S3PP platform, an escalated identity revelation system and trusted shared storage (EIRS & TSS). It discusses the architectural, structural, and behavioral view of the system through which the functional and non-functional requirements are implemented.
The intended audience of this document are the software architects, engineering, developers, and all the stakeholders who may be interested in the software design of the system.

1.2 Scope

S3PP is composed of an escalated identity revelation system and a trusted shared storage (EIRS & TSS). Moreover, it offers a social platform by allowing its users to create multiple identities with different levels of identity revelations, and then use them to interact with each other by sending messages, creating group chats, going live, writing to their blogs, and posting photographs. It also offers the users a file-storage-and-sharing facility to create, edit, read or share files in a secure manner through means of encryption.

1.3 Overview

This document consists of four main parts. The first part describes the architectural structure of the system by giving a full description of the architectural model, the subsystems of the system and how they interact with each other and with the environment. The second part shows the structural design by describing the class diagram of the system including each class, its attributes and methods, and how they interact with each other via different controllers. Controller classes are discussed in detail as well. The third part of the document describes the data flow and data dictionary of the system. The last part describes the design of the user interface by showing screenshots of the screen and how its objects interact with each other and with the end-users.

2. System Overview

The system of S3PP is composed mainly of four main subsystems. Each is designed to deliver a specific feature. First, the identity tracking system (ITS) which is designed to allow the user to maintain all his identities. Moreover, it allows the user to allocate the bots that could impersonate each identity through the bot store. The second system is the social system which mainly represents a platform on which the users could do their social activities such as text, video and audio chatting, writing blogs, going live, posting photos and getting recommendations. The third system is the payment system; it is mainly designed to coordinate the delivery and processing of the payments between the user and the content creator. Finally, the last system is the trusted shared storage (TSS) which is designed to make users able to control their files in terms of creation, deletion, editing, sharing and encryption. Figure 1 shows the subsystems of S3PP system.

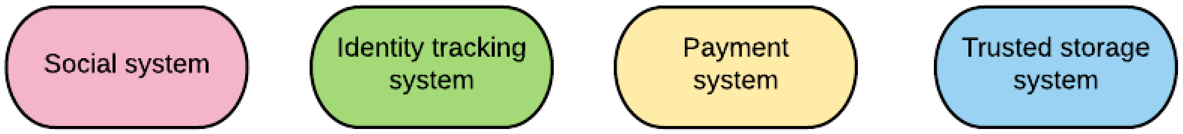


Figure 1

3. System Architecture

3.1 Architectural Design

Figure 2 shows the architecture of S3PP system.

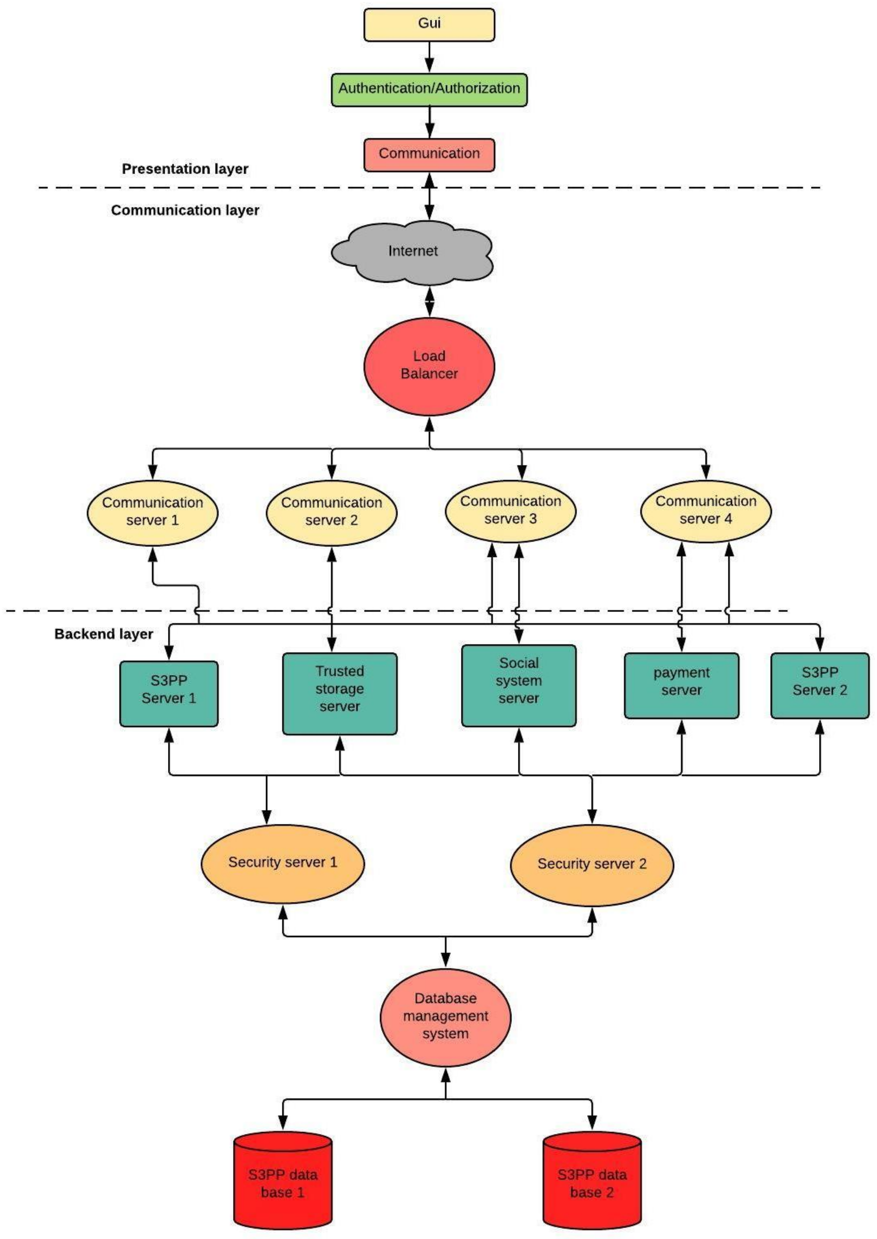

Gui
Authentication/Authorization
Communication
Presentation layer
Communication layer
Internet
Load Balancer
Communication server 1
Communication server 2
Communication server 3
Communication server 4
Backend layer
S3PP Server 1
Trusted storage server
Social system server
payment server
S3PP Server 2
Security server 1
Security server 2
Database management system
S3PP data base 1
S3PP data base 2

Figure 2

The architecture of the system is generally a mixture between a layered architecture and a client server architecture as per the following:

1. Presentation layer:
    a. GUI: The graphical user interface is a form of user interface that allows users to interact with the platform through graphical icons and audio indicators such as primary notation, instead of text-based user interfaces, typed command labels or text navigation. The user uses this layer to log into the system and interact with the rest of the features we provide. The inputs of the user in this layer are to be forwarded to the following layers.

    b. Authentication/Authorization : This layer is responsible for authenticating the user's inputs and requests. This layer encrypts the data that was sent by the user to the server side through the relatively insecure network (internet) then it decrypts the relevant data that was sent from the servers through the internet. It provides access control for systems by checking to see if a user's credentials match the credentials in a database of authorized users or in a security server, then It authorizes the users according to their user class to the available features.

    c. Communication: This layer mainly manages the data transfer and provides proper communication between the user and the server side. It generates the responses needed for the communication between the two sides.

2. Communication layer:
    a. Internet: This layer allows millions of users to access the platform anytime and have a proper way of connecting to each other. It is better for the user to access the different services that we provide on our platform and be updated about what is new on the platform and what is being announced. It acts as the media that transfers the users' requests to the server side.

    b. Load Balancer: The function of this component is that it distributes client requests and routes them to a communication server in a manner that maximizes speed and ensures that no one server is overworked. Moreover, if one communication server is down, the load balancer automatically redirects the client requests to the online server. Furthermore, it allows new servers to be easily added to the system.

    c. Communication servers: Communication servers receive the requests from the client (that were directed before by the load balancer), then these requests are routed back to the relevant server (request capturing). Moreover, these servers generate responses and send them to the client side over the internet. We have built four communication servers to increase the efficiency and the speed of our system by dividing the requests between these communication servers such that none of them is overloaded. In addition, if one of them is down, the other one takes over and the overall system does not crash.

3. Servers layer:
    a. S3PP server: S3PP Server handles all the operations of the application that is related to the Identity processes and operations and the bots store as well. This server is duplicated as this will be used more than the social server, the payment server and the TSS server as it has to do with more operations, so in order to eliminate the risks of fall time or server overload.

    b. Payment server: it handles payments requests, all the transactions information, payments dates, delays, and amounts retrievals or requests.

    c. Social system server: This server handles all the chat operations. If a client decides to create a new chat, the API is called. If the client decides to send a message, the API is called. If a client receives a message, the API is called. Even if a client wants to delete a message, the API is called. Therefore, the communication server redirects all chatting requests to this server, and it calls the API, awaits a response from the API, then acts according to the received response. Moreover, it handles the user’s blogs, going live, viewing recommendations and posting content.

    d. Trusted storage server: This server is responsible for the creation, deletion, modification and encryption of the files and content shared on the system by making use of the file creation identity (FCD).

4. Security server: This layer encrypts the data before storing it in the database to be more secure. Moreover, it encrypts the data before sending it over the insecure network (internet) to the client side and we have two of it in order to enhance the availability and performance of the system.

5. Database Management System:
   This layer is responsible for all database operations, whether inserting, deleting or updating. Any other layer or server that needs to access the database should do so through this layer. This layer is responsible for generating the SQL queries needed to do the required operation on the database.

6. Database Server:
   The database is a central database similar to the repository architecture. However, there are two database servers as the database is replicated and backed up to be used in the condition of a database failure. The two databases are simultaneously updated.

3.2 Decomposition Description

S3PP system architecture is composed of four main layered subsystems; each of which consists of different underlying components, as follows:

1. Presentation Layer:
   a. GUI:
   The Graphical User Interface allows user interaction with the system through graphical icons and audio forms using various input components. Figure 3 is a decomposition of the GUI layer.

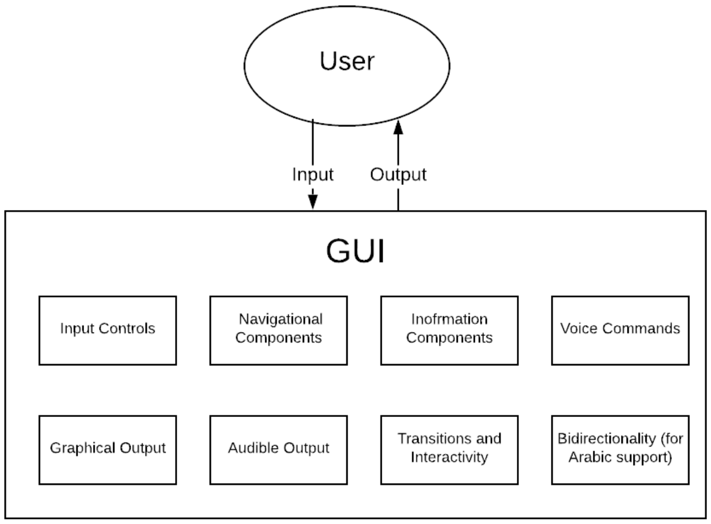


Figure 3

   b. Authentication:
   The authentication layer is responsible for validating users' input, preventing incorrect input, and explaining types of valid input forms whenever possible. Figure 4 shows a decomposition of the authentication layer.

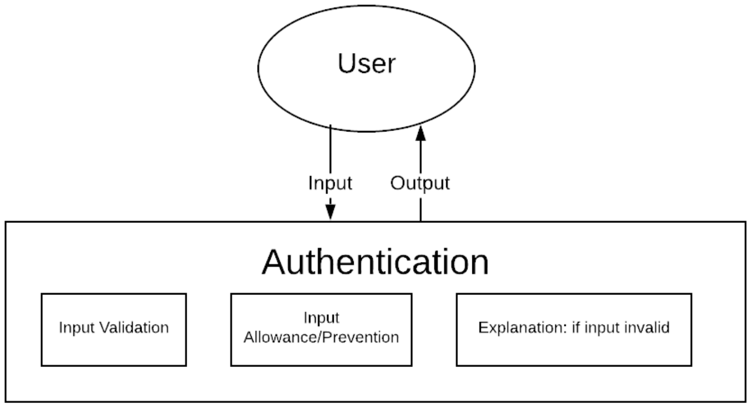


Figure 4

c. Communication:
The communication layer is the middleware layer between the user interface and the underlying communication and server layers by controlling data transfer, encryption, and decryption, or translation, between both sides.

2. Communication Layer:
   a. Internet:
   The internet layer is responsible for providing internetworking, moving packets from source to destination, and the transmission and delivery of messages between processes on source systems to destination systems, and thus allowing access to network resources.

   b. Network Load Balancer:
   The network load balancing layer is responsible for splitting and distributing the internet traffic over multiple internet connections for increased internet performance, reliability, and connectivity.

   c. Communication Server:
   The communication server layer is responsible for providing communication services for users on the network who need to transfer files or access information on the systems or networks at remote locations over telecommunication links.

Figure 5 shows a decomposition of the underlying subsystems of the communication layer.

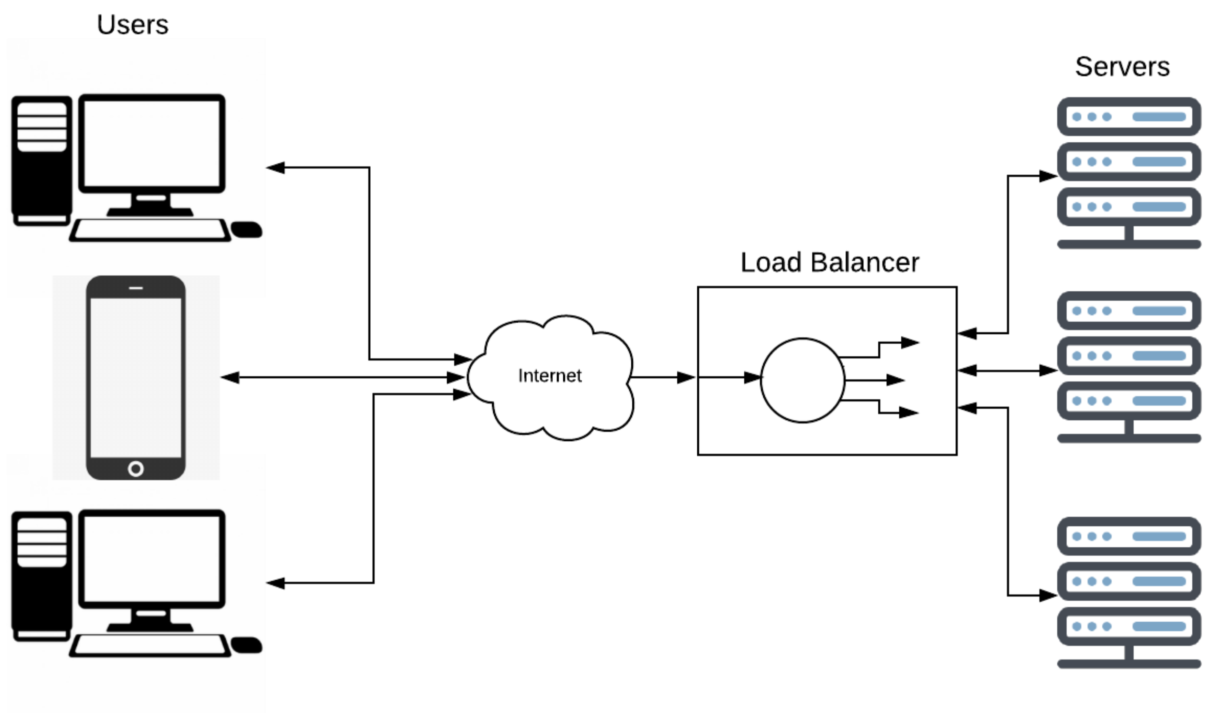


Figure 5

3. Backend Layer:
   a. Application System Servers:
      i. S3PP Servers:
         The S3PP servers encompasses many of the application operations a user could request as well as the bots-specific operations.
         A decomposition of an S3PP server is shown in Figure 6.

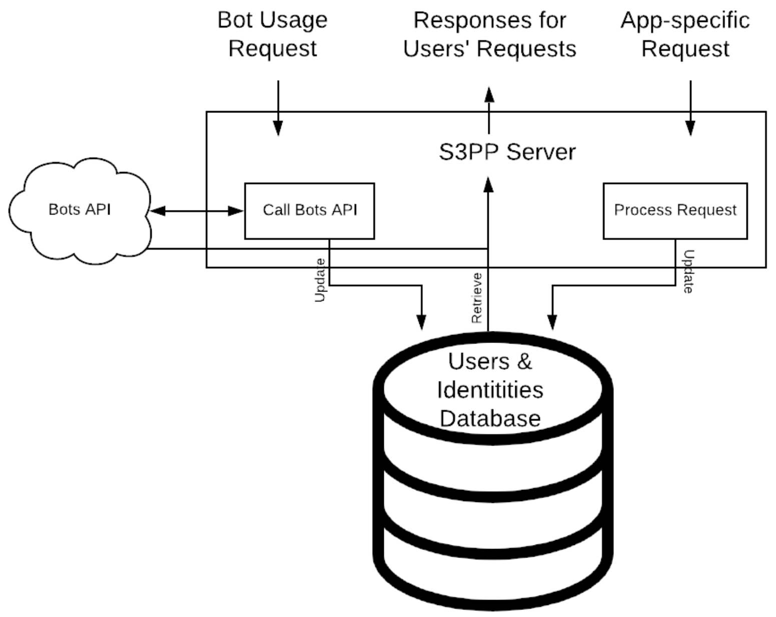


Figure 6

ii. TSS Server:

The TSS server layer is responsible for files management such as files creation, editing, deletion, and for sharing of files if the same FCD is assumed. A decomposition of the TSS server is shown in Figure 7.

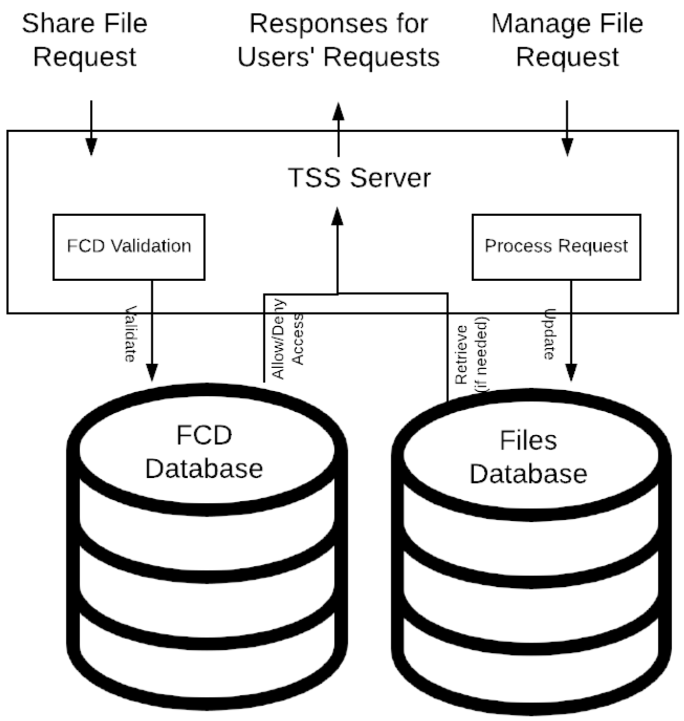


Figure 7

iii. Social System Server:

The social systems server layer is responsible for the chatting and the different social operations a user could use. A decomposition of the Social System server is shown in Figure 8.

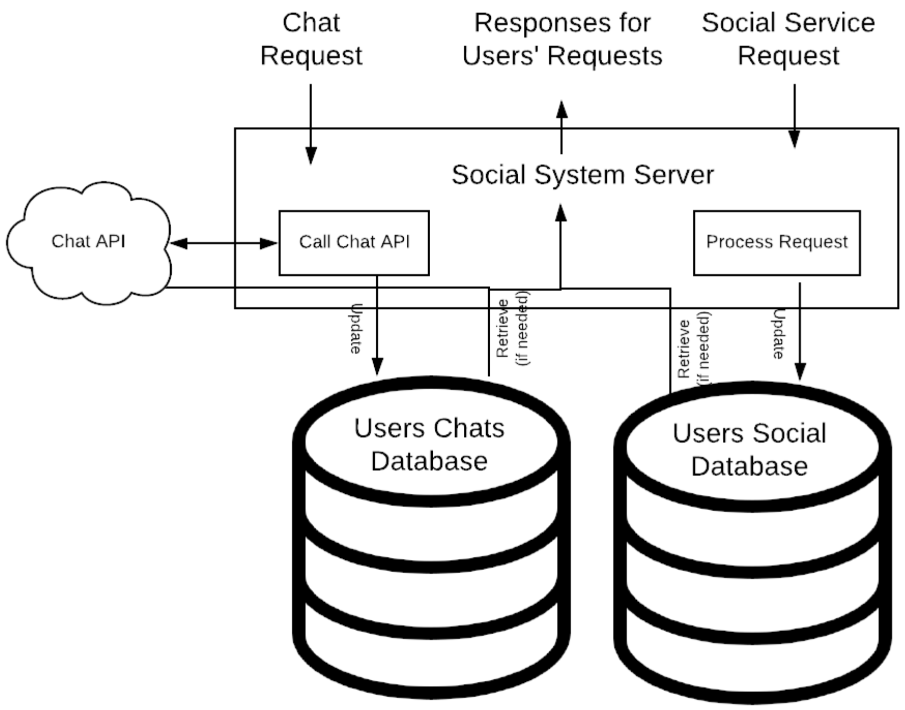


Figure 8

iv. Payment Server:

The payment server layer is responsible for handling the whole process of all the users' transactions from the payments' requests until the corresponding output. Figure 9 shows a decomposition of the payment system layer.

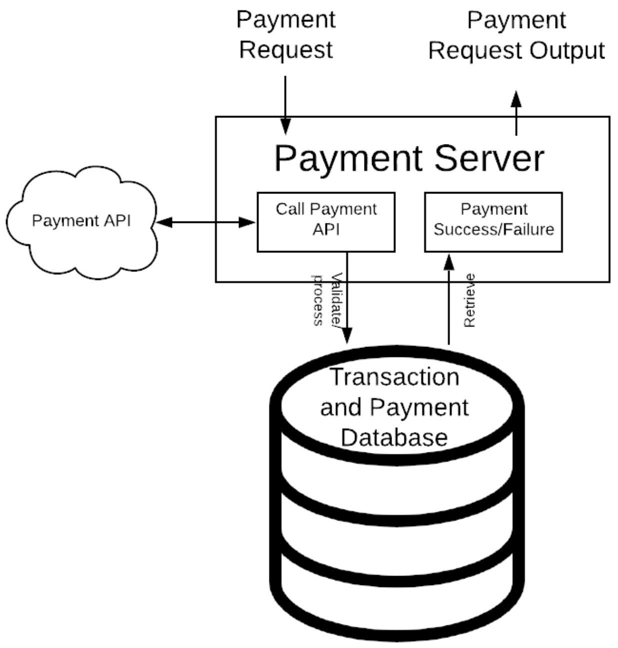


Figure 9

b. Security Servers:
   The security servers layer ensures the protection of information by encrypting the input data and decrypting the output data from the underlying database layer.
c. Database System:
   The database layer system is an organized collection of all types of data needed to be accessed or stored by the user and for the application systems. Figure 10 shows a decomposition of the security servers and the database system layers.

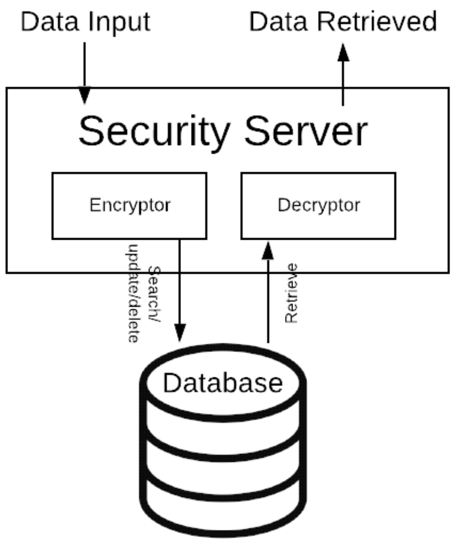


Figure 10

3.3 Design Rationale

The system as a whole is implemented as a client-server architecture. This was chosen because the application is implemented as a web/mobile application, so it should not take too much space on the user’s device. Therefore, most of the components and the database of the system are present at the server side and its interaction takes place through the internet. Moreover, since applications are difficult to change, adding business rules or processes to the application does not facilitate business agility. This is where layered architecture comes into play. It is widely adopted and used by software architects, designers, and developers. The layered architecture pattern closely matches the conventional IT communication and organizational structures found in most businesses.

The client-server architecture has several other benefits. For instance, the user can use the application despite his/her location and the device that they are using. Furthermore, security is ensured as a client must be authenticated and authorized to be able to take advantage of the functions and the data processing capabilities of the system. From a maintainability perspective, client-server architectures are maintainable as the server can be replicated on another computer and then the server can be easily replaced or re-allocated, without the client knowing about that.

Yet, the layered architecture has its drawbacks. Some of which are:

- Lack of inbuilt scalability:
  The principles of layered architecture are not suitable for scaling the project.
- Hidden use cases:
  It is difficult to determine the use cases by simply checking the code organization. We need to refer to the class names and in most cases, implementation.
- No dependency inversion:
  In a layered architecture, the dependencies are direct and conceptually changed into essential higher layers from a low-level infrastructure layer.
- Lesser Performance:
  Lesser performance is due to the extra overhead of passing through layers instead of calling a component directly.
- Overloaded servers.

We overcome this by duplicating the servers which will be extensively used and making use that the load balancer distributes the workload accordingly.

The client side and the server side, each, are implemented as a separate layered architecture. This is because layered architecture allows each layer to be developed, tested, deployed, maintained and updated separately. It also allows that layers can be added, removed or changed without affecting the system as long as the same interface is provided. Furthermore, it allows us to have different levels of security for different layers. Hence, with all of advantages and drawbacks in mind, we found that the mixture of layered and client-server architectures was a suitable and efficient mix to implement.

4 Data Design

4.1 Data Description

Figure 11 shows the class diagram of the S3PP system.

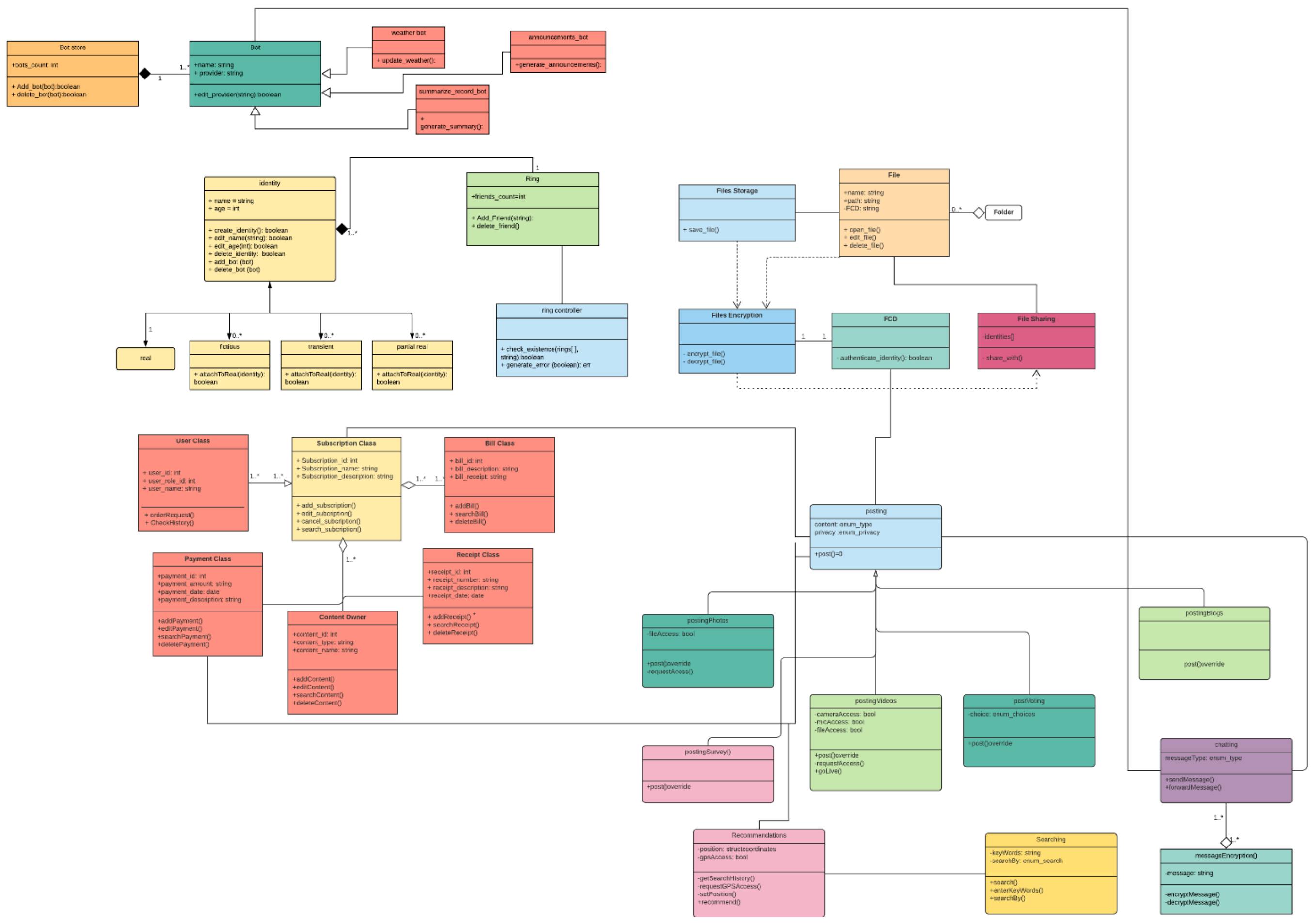


Figure 11

4.2 Data Dictionary

Class Name	Identity	
Class ID	1	
Attributes	-name: string	The name of this identity
	-age: int	The age the user assumes in this identity
Methods	- create_identity(): boolean	This method creates the identity object. The boolean would indicate the success or failure of the operation.

Class Name	Identity	
Class ID	1	
	edit_name(string): boolean	This method edits/sets the identity’s name.The boolean would indicate the success or failure of the operation.
	edit_age(int): boolean	This method edits/sets the identity’s age.The boolean would indicate the success or failure of the operation.
	delete_identity: boolean	This method deletes the identity’s object and calls the destructor.
	add_bot (bot)	This method adds a bot and makes it able to impersonate the user’s identity.
	delete_bot (bot)	This method deletes a bot and disassocites it from the following identity.

Class Name	Ring	
Class ID	2	
Attributes	friends_count: int	The number of friends in a ring.
Methods	Add_Friend()	This function adds a friend to the identity’s friend list.
	delete_friend()	This function deletes a friend from the identity’s friend list.

Class Name	Ring Controller	
Class ID	3	
Attributes		
Methods	check_existence(rings[ ], string):boolean	This function takes all the rings associated with a real user's identities. The boolean would indicate the success or failure of the operation.
	generate_error (boolean): err	This function returns an object of type error that represents that a friend exists in another ring.

Class Name	Real
Class ID	4
Attributes	Inherits the Identity Object class attributes
Methods	Inherits the Identity Object class methods.

Class Name	Fictitious
Class ID	5
Attributes	Inherits the Identity Object class attributes
Methods	Inherits the Identity Object class methods in addition to:

Class Name	Fictitious	
Class ID	5	
	attachToReal(identity): boolean	This function attaches this identity to the real user's identity.

Class Name	Transient-fictitious	
Class ID	6	
Attributes	Inherits the Identity Object class attributes	
Methods	Inherits the Identity Object class methods, in addition to:	
	attachToReal(identity): boolean	This function attaches this identity to the real user's identity.

Class Name	Partial Real	
Class ID	7	
Attributes	Inherits the Identity Object class attributes	
Methods	Inherits the Identity Object class methods, in addition to:	
	attachToReal(identity): boolean	This function attaches this identity to the real user's identity.

Class Name	Bot Store	
Class ID	8	
Attributes	bots_count: int	The number of available bots in the store.
Methods	add_bot(bot):boolean	Add a bot object to the store. The boolean would indicate the success or failure of the operation.
	delete_bot(bot):boolean	Delete a bot object from the store. The boolean would indicate the success or failure of the operation.

Class Name	Bot	
Class ID	9	
Attributes	bot_name: string	The name of the bot.
	bot_provider: string	The name of the bot's provider.
Methods	add_bot(bot):boolean	Add a bot object to the store. The boolean would indicate the success or failure of the operation.
	delete_bot(bot):boolean	Delete a bot object from the store. The boolean would indicate the success or failure of the operation.

Class Name	Weather Bot	
Class ID	10	
Attributes	Inherits the Bot Class attributes.	
Methods	Inherits the Bot Class methods in addition to:	
	update_weather()	This function gives the user the recent weather updates.

Class Name	summarize_record_bot	
Class ID	11	
Attributes	Inherits the Bot Class attributes.	
Methods	Inherits the Bot Class methods in addition to:	
	generate_summary()	This function gives the user the summary of the conversations and alerts the user to important announcements.

Class Name	announcements _bot
Class ID	12
Attributes	Inherits the Bot Class attributes.
Methods	Inherits the Bot Class methods in addition to:

Class Name	announcements _bot	
Class ID	12	
	generate_ announcements()	This function stores and announces events for a certain forum.

Class Name	File	
Class ID	13	
Attributes	file_name: string	The name of the file.
	file_path: string	The path of the file.
	FCD: string	The file creation identity with which the file is associated.
Methods	open_file()	This function opens the file with the read-only attribute.
	edit_file()	This function allows write access to an opened file or a file to be opened.
	move_file()	This function moves the file to a new file location.
	copy_file()	This function copies the file to a new file location.
	delete_file()	This function deletes a file the user has created.

Class Name	Folder
Class ID	14
Attributes	Inherits the File class attributes
Methods	Inherits the File class methods.

Class Name	File Cryptography	
Class ID	15	
Attributes	Inherits the File Class attributes in addition to	
	public_key: string	The key used for file encryption.
	private_key: string	The key used for file decryption.
Methods	encrypt_file()	This function encrypts a file using a public key.
	decrypt_file()	This function decrypts an encrypted file using a private key.

Class Name	FCD
Class ID	16
Attributes	Inherits the File Cryptography Class attributes.

Class Name	FCD	
Class ID	16	
Methods	authenticate_identity(FCD): boolean	This function authenticates whether or not the user identity assumes the same FCD as that associated with the file in process.

Class Name	File Sharing	
Class ID	17	
Attributes	Inherits the File Class attributes in addition to	
	Identities[]: boolean	A list of the identities with which the user chooses to share a file.
Methods	share_file()	This function shares a file with a number of user identities.

Class Name	Payment	
Class ID	18	
Attributes	sender: string	The username of the payment sender.
	description: string	A brief description of the payment is for.
	date: dateType	A log for previous payments' dates.

Class Name	Payment	
Class ID	18	
	receiver: string	The username of the payment receiver.
	amount: int	The payment amount between the sender and the receiver.
Methods	pay()	This function is responsible for the payment process and is called whenever the user wants to make a transaction.
	search()	This function searches for previous payments and shows a log of these payments.
	set_amount()	This function is used to set the amount to be paid by the user whenever a transaction is taking place.
	get_amount()	This function saves the amount paid by the user in the payment and transactions database.

Class Name	Content	
Class ID	19	
Attributes	ID: int	The ID associated with a given content.
	type: string	The type of the content.
	owner_name: string	The content owner’s name.

Class Name	Content	
Class ID	19	
Methods	add(name, ID)	This function inherits from the subscription class and is called when an owner wants to put up content.
	edit(name, ID)	This function is used by the owner if they need to edit something, such as a price, in the content they uploaded.
	delete(name, ID)	This function is used to delete a content.
	search(name, ID)	This function is used to search for content.

Class Name	Receipt	
Class ID	20	
Attributes	receipt_ID: string	The receipt ID.
	amount: int	The amount of money paid.
Methods	search(name, ID)	This method is called whenever the user decides to check previous payments.
	delete(name, ID)	This function is used for deleting old receipts.

Class Name	Bill	
Class ID	21	
Attributes	bill_ID: int	The ID of the bill sent to the user before payment.
	amount: int	The amount of money paid.
	description: string	A description of the content requested.
Methods	search(name, ID)	This method is called whenever the user decides to check previous payments.
	delete(name, ID)	This function is used for deleting old receipts.

Class Name	Subscription	
Class ID	22	
Attributes	ID: int	The ID associated with the subscription.
	name: string	Identification of the subscription.
	description: string	A description of the content requested.
Methods	add(name, ID, description)	This method is called whenever the content owner wants to add a new subscription.

Class Name	Subscription	
Class ID	22	
	Edit(name, ID, description)	This function is used to edit previous subscriptions the owner has put up.
	cancel(name)	This is used by the owner of the content or the user who is paying for it to terminate that subscription either stop paying if it is the user or remove it completely by the owner.
	search(ID)	This function is used to search for certain subscriptions.

Class Name	Paying User	
Class ID	23	
Attributes	ID: int	The ID associated with the user under which subscription.
	name: string	The name of the user.
	Role: int	The role of the user.
Methods	order_request(name)	This method is used by the user when they find some content they want to make available for themselves. The function is invoked when the user makes such request. The function inherits from the subscription class.

Class Name	Paying User	
Class ID	23	
	check_history()	This function is used by the user to check previous subscriptions to ease re subscribing to that content if it had a time limit.

Class Name	Posting	
Class ID	24	
Attributes	-type: enumtype	A variable that can take a value of { blog, photo, video, live, voting or survey}
	+privacy: enumPrivacy	A variable that can take a value from {public, ring, only me}
Methods	+post()=0	A virtual function that takes the default value of 0 and responsible for publishing new content

Class Name:	postingPhotos	
Class ID	25	
Atrtributes	-fileAccess: bool	This variable holds the value of 1 if the app is given the permission to open the file explorer and 0 otherwise.
	-photo: structPhoto	A struct that holds the dimensions and information about the photo.

Class Name:	postingPhotos	
Class ID	25	
Methods	+post()override	This is an overriding function for the post method in the parent class posting. This function is responsible of uploading and posting photos whenever it is granted the access (when fileAccess=1). Then it set the value of fileAccess to 0 if it is not given permanent access.
	-requestAccess()	A utility function that is called in post() if the value of fileAccess is 0 to change its value to 1 if access is granted.

Class Name:	postingVideos	
Class ID	26	
Attributes	-cameraAccess: bool	This variable holds the value of 1 if the app is given the permission to open the camera and 0 otherwise.
	-micAccess:bool	This variable holds the value of 1 if the app is given the permission to open the mic and 0 otherwise.
	-fileAccess:bool	This variable holds the value of 1 if the app is given the permission to open the file explorer and 0 otherwise.
	-video: structVideo	A struct that holds the information about the uploaded video

Class Name:	postingVideos	
Class ID	26	
Methods	+post()override	This is an overriding function for the post method in the parent class posting. This function is responsible of uploading and posting videos whenever it is granted the access (when fileAccess=1). Then it set the value of fileAccess to 0 if it is not given permanent access.
	+goLive()	This function is responsible of live streaming. It can work whenever both micAccess and fileAccess are set to 1
	-requestMicAccess()	A utility function that is called in goLive() if the value of micAccess is 0 to change its value to 1 if access is granted.
	-requestCameraAccess()	A utility function that is called in goLive() if the value of cameraAccess is 0 to change its value to 1 if access is granted.

Class Name	postingSurvey	
Class ID	27	
Attributes		
Methods	+post()override	Overriding function to post a survey

Class Name	postingBlogs	
Class ID	28	
Attributes		
Methods	+post()override	Overriding function to post a blog

Class Name	postingVoting	
Class ID	29	
Attributes	-choice: enum_choices	A variable that can take values that are automatically added to the enum called enum_choices
Methods	+post()override	Overriding function that gives a value to the variable choice

Class Name	Chatting	
Class	30	
Attributes	-messageType: enum_type	A variable that can take a value from the enum_type that can be a video, text , photo or a voice note
Methods	+sendMessage()	A function that is used to send the message of whatever type to the receiver.
	+forwardMessage()	A specific version of the send message function that is responsible of sending the selected message to another user out of this chat.

Class Name	Recommendations	
Class ID	31	
Attributes	-position: structCoordinates	This struct holds the x and y coordinates of the current position of the device
	gpsAccess: bool	The variable holds 1 if the app is allowed to open the GPS and 0 otherwise
Methods	-getSearchHistory()	A friend function that access the database to find the search history of the user to be able to recommend
	-requestGPSAccess()	A utility function that is called to ask for permission to open GPS if the current value of gpsAccess is 0
	setPosition()	A setter function that uses the GPS to set the value of the variable position
	+recommend()	This function can be called to recommend functions or pages to like according to the parameters passed to it. 1 means pages and 0 means restaurants

Class Name	searching	
Class ID	32	
Attributes	-keyWords: string	A variable that stores the input the user entered to be searched for
	-searchBy: enum_search	A variable that can take a value of type place or name

Class Name	searching	
Class ID	32	
Methods	+enterKeyWords()	A function that allows the user to set the value of the variable keyWords
	+searchBy()	A setter function that helps the user to choose the method of search
	+search()	A function that gives the order of searching and it access the database of the app to get the permissible content

Class Name:	**messageEncryption**	
Attributes	-message: string	A variable that holds the string of bits that shall be encrypted
Methods	-encryptMessage()	A function that encrypts the message with end-to-end encryption
	-decryptMessage()	A function that uses a key to decrypt the message from the receiver side

5. Human Interface Design

5.1 Overview of User Interface

The interaction between the end users and the system will be through different user interfaces as the end users will be allowed to utilize the system's functionalities through the following actions:

1. Sign up for a new account if it is not yet.
2. Login into their created account

3. Add another identity whether transient fictitious, fictitious, partial or real.
4. Delete an identity
5. Switch between different identities.
6. Accept friend requests.
7. Add/remove/block friends in his/her ring.
8. Assign a bot to a specific identity.
9. Buy new bots from a third party and add them into their desired identities
10. Take summary of the important conversations through Silently Summarize And Record Bot.
11. Have notifications about the weather through the weather bot.
12. Receive announcement about the important events and forums through announcement bot.
13. Store/share files
14. Create/read/edit/share/delete files after entering a certain FCD.
15. Chat (text/ video / audio) with other friends.
16. Create new group chat.
17. Add personal blogs.
18. Like/dislike blogs /photos.
19. View profiles of other contacts.
20. Search for other friends.
21. Create photo albums.
22. add/delete/hide photos.
23. Track physical locations.
24. Lookup and receive recommendations for eg: good restaurants.

25. Do surveys and votings.

26. Go live and audio streaming.

27. Pay for the service.

28. Paying participants can get additional support services.

    a. Paying participants can get advertisement free experience.
    b. Get paid for participation if you are a celebrity.

29. Get free services.

30. Receive and view advertisements.

The following table explains what feedback the system gives to the user whenever an action occurs.

Action	System feedback
1. Sign up for a new account if it is not yet.	The interface displays a form to the users to submit their data. Then the system will insert those data into its database and display a success message for this process.
2. Login into their created account	The system will compare the entered user/password information with the existing data in its database then display one of two options: 1- redirect the user to his/her ITF user interface in case of entering correct info.

	2- displaying an error message to reenter the username/password.
3. Add another identity whether transient fictitious, fictitious, partial or real.	The interface displays a select option button for the user to choose his/her desired identities, then the result will be added to the database and displayed into his/her ITF system.
4. Delete an identity	The interface will delete the selected identity with its bots from the user profile and from its database as well.
5. Switch between different identities	The interface will display a page showing the different existing identities for the user based on his entered data previously and ask her/him to choose only one to make it active for him into the profile.

6. Accept friend requests.	The interface will send a notification for the user with a friend request combined with a link to this person's profile to check who is that. In case the user accepts this request, the interface will send a notification for that friend saying that his request has been accepted and gives that friend the authority to access the user's profile.
7. Remove/block friends.	The interface will display two options for the user, either to block or remove it, In case the user removes that friend, the interface will remove the info for this friend into the database and remove it from the user's ring associated with this specific identity.
8. Assign any of their identities to their existing bots.	The interface will display a screen showing the available bots inside the user account, and this bot will be added to his account identity.
9. Buy new bots from a third party and add them into their desired identities	The interface will provide the users to the bot store where they can add/delete bots either provided by S3PP or third parties. They can associate them with a specific identity in the ITF.

10. Take a summary of the important conversations through Silently Summarize And Record Bot.	The interface will send a message to the user with highlights of the important conversations.
11. Have notifications about the weather through the weather bot.	The interface will display a widget with the expected weather for the user.
12. Receive announcement about the important events and forums through announcement bot.	The interface will send notifications with the dates of the important forums and events to the user.
13. Store/share files	The interface will display a page to allow the user to upload his/her files from computers and store them into a cloud. It will display a button for sharing those files with people.
14. Create/read/edit/share/delete files after entering a certain FCD	The interface will display a form first for the user to enter a specific FCD shared between two users or the user itself, then the interface will validate these info and in case of succeeding in receiving the correct FCD, it opens a blank page for the user to write/ read the files.
15. Chat (text/ video / audio) with other friends.	The interface will display the sent and received messages history.

16. Create new group chat.	The interface will display a button for starting a new chat with other friends in the ring.
17. Add personal blogs.	The interface will display a text box for the user to upload/ write his personal blogs.
18. like/dislike blogs/photos.	The interface will display one button for like and dislike. In the case of a like, the interface will send a notification for the person’s post saying that the X user liked your blog/photo.
19. View profiles of other contacts.	The interface will display the profile for a certain friend.
20. Search for other friends.	The interface displays a search bar for the user to search for other contacts.
21. Create photo albums.	The interface will display a button for the user to upload his own photos.
22. add/delete/hide photos.	The interface will display three options for the user to add/delete or hide photos from other contacts in his/her rings.

23. Track physical locations.	The interface will allow tracking systems like GPS to track the physical locations.
24. Lookup and receive recommendations for example: for good restaurants and shops.	The interface will send notifications for the user with specific recommendations for restaurants or mainly what he liked.
25. Do surveys and votings.	The interface will send to the user a link to google form to submit some surveys.
26. Go live and audio streaming.	The interface will display a Go live button.
27. Pay for the service.	The interface will display a page for the user to enter his/her credit card information, the method of paying and the available options for him/her.
28. Get paid for participation if you are a celebrity.	The interface will send a mail for the celebrities with the specific information required to get paid.
28. Get free services	The interface will display all the available free services from which the user could use.

29. Receive and view advertisements.	The interface will display some advertisements for the users who participated in the service.

5.2 Screen Images

1. Welcome Screen

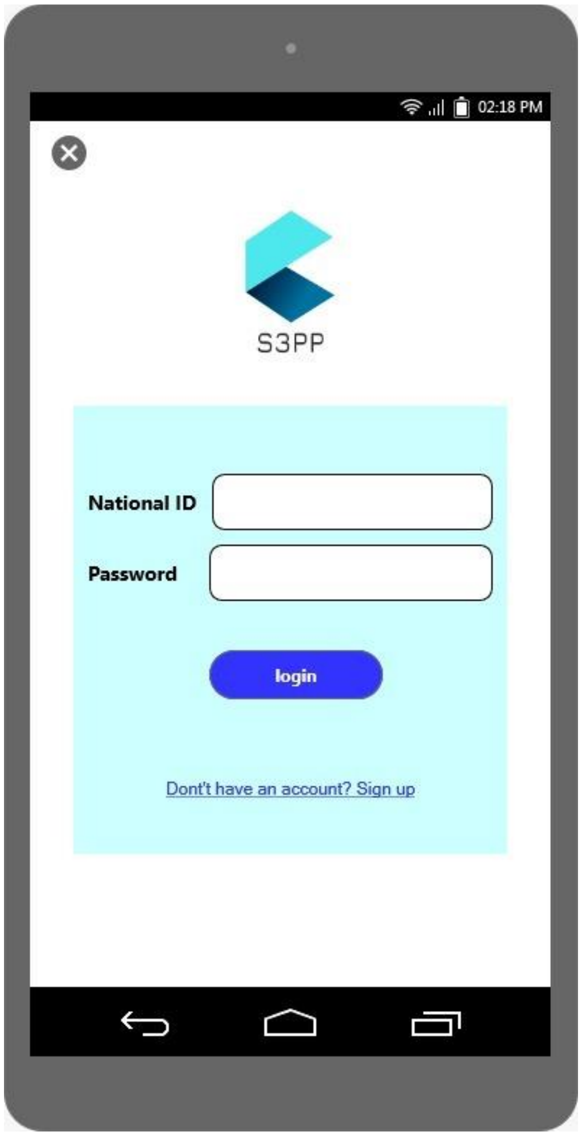
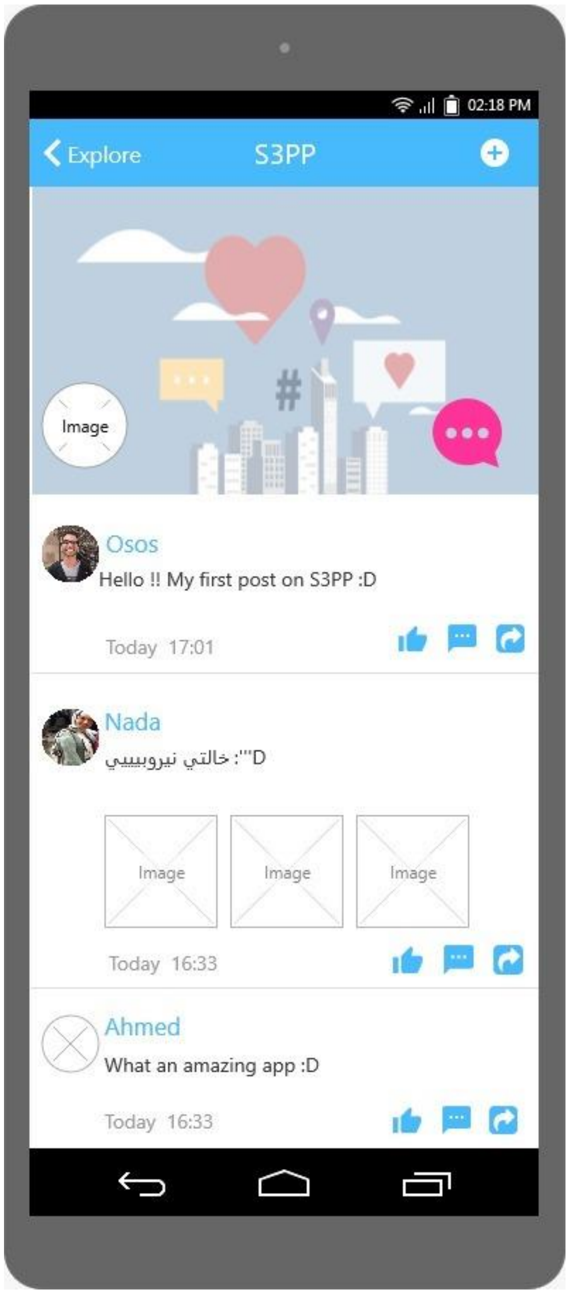


Figure 12

Figure 12 shows the first screen that is being displayed whenever the user opens the app. The user can:

1. Log in by entering his national ID and password and then press the button login (1).
2. Sign up for a new account by clicking the hyperlink Don't have an account? Sign up (2).

2. Explore Screen



Hello !! My first post on S3PP :D

خالتي نيروبييي :'''D

What an amazing app :D

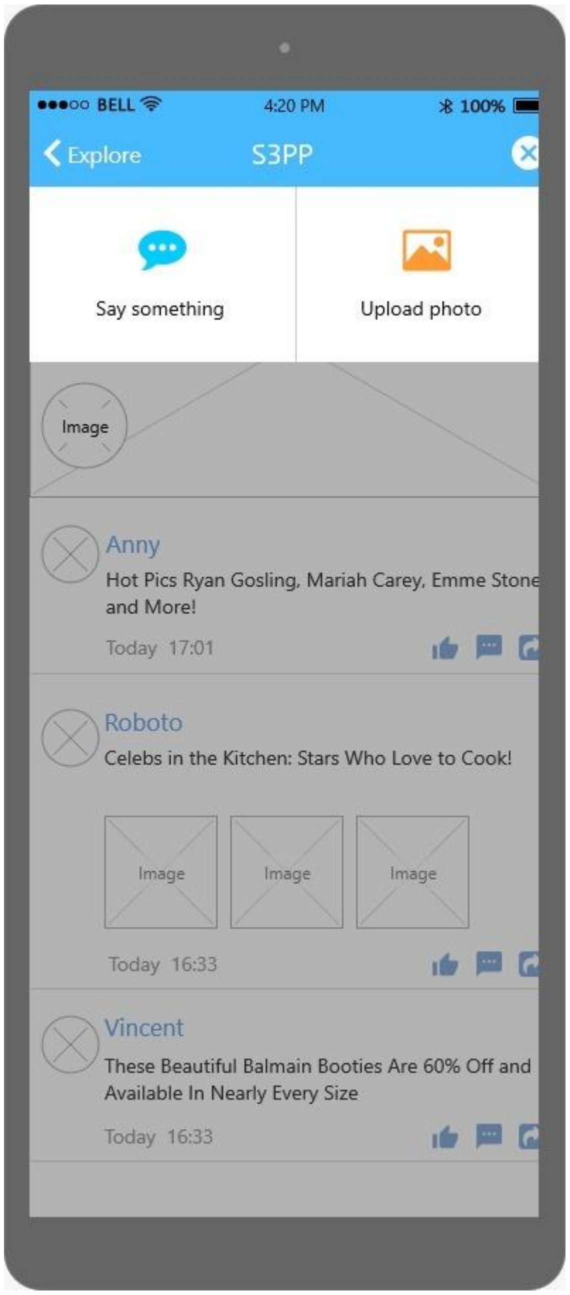
Figure 13

In the screen in Figure 13, the user can do either of the following:

1. Keep browsing the posts and interact with them by Like (1), Comment (2) or Share (3).
2. Post new content by hitting the icon + (4).

3. Back one step by clicking the back icon (5).
4. Send a message by clicking the red icon message (6).

3. Posting Screen



Anny
Hot Pics Ryan Gosling, Mariah Carey, Emme Stone and More!
Today 17:01

Roboto
Celebs in the Kitchen: Stars Who Love to Cook!
Today 16:33

Vincent
These Beautiful Balmain Booties Are 60% Off and Available In Nearly Every Size
Today 16:33

Figure 14

This screen shown in Figure 14 is displayed when the user clicks the + icon. In this screen the user can do either of the following:

1. Write down something using the button Say something (1).
2. To post a photo using the button upload photo (2).

3. Close the prompt using button back and then the user is redirected to screen Explore (3) that allows the user just to browse the posted content.
4. Close the prompt using x (4) icon.

4. Chatting Screen

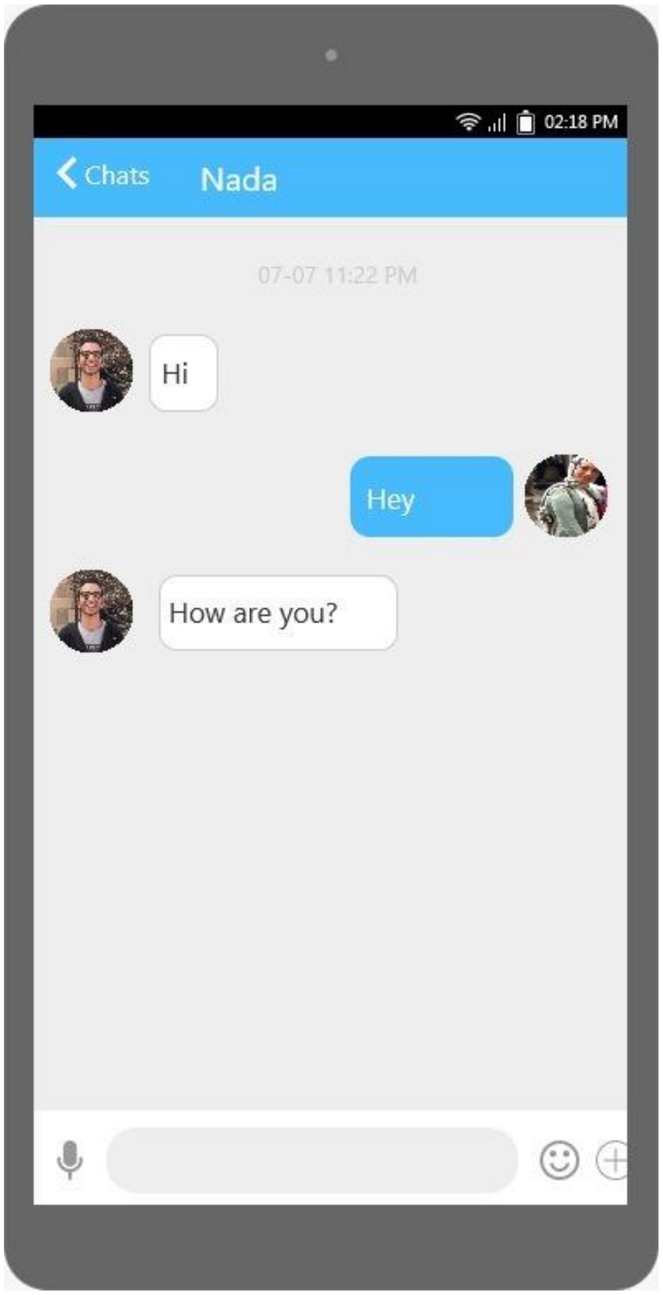


Figure 15

In the screen in Figure 15, the user can send any of the following:

1. Send a text message by touching the text box (1).
2. Send an audio recording by touching the Mic icon (2).

3. Write an emoji by touching the emoji icon (3).
4. Upload a photo or video by touching the + icon (4).

5. Identity Management Screen

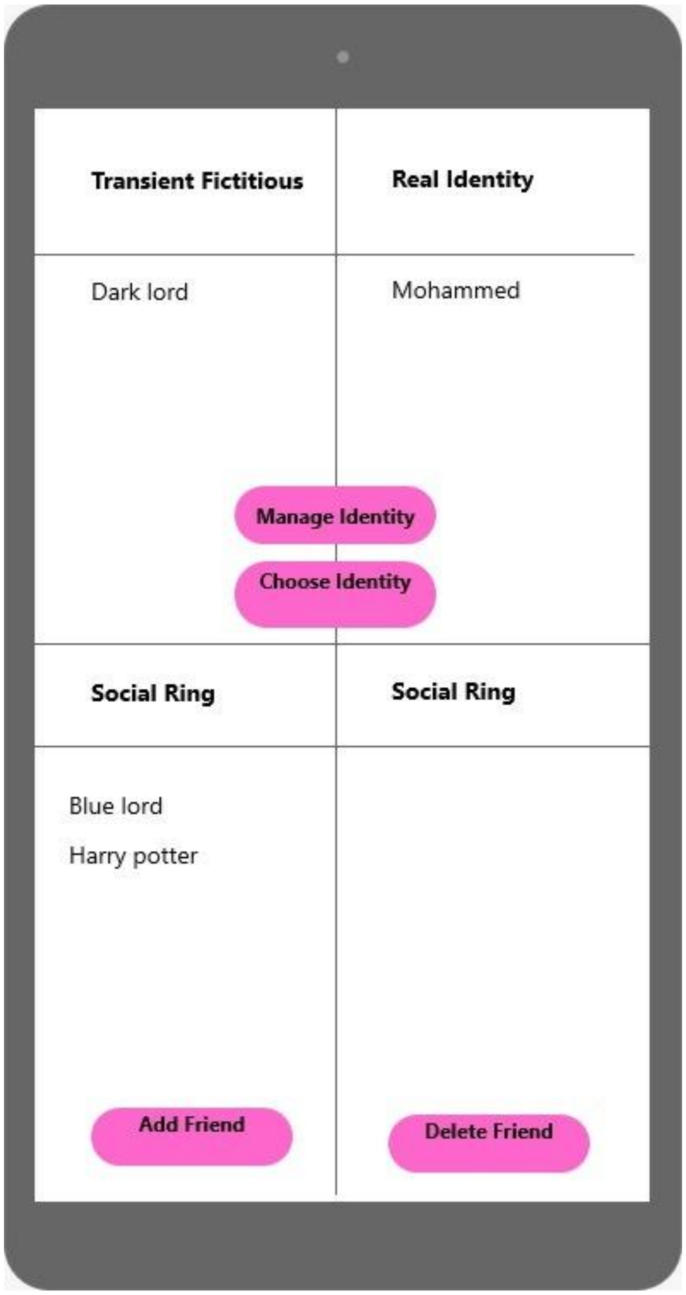


Figure 16

In the screen shown in Figure 16, the user can manage his identities and perform the following:

1. Manage an identity by clicking the Manage Identity button (1).
2. Add a friend to a specific ring by hitting the button Add Friend (2).
3. Unfriend somebody by hitting the Delete Friend (3) button.

4. Choose an identity to continue using the app with this this identity using the Choose Identity (4) button.

6. Publishing Content Screen

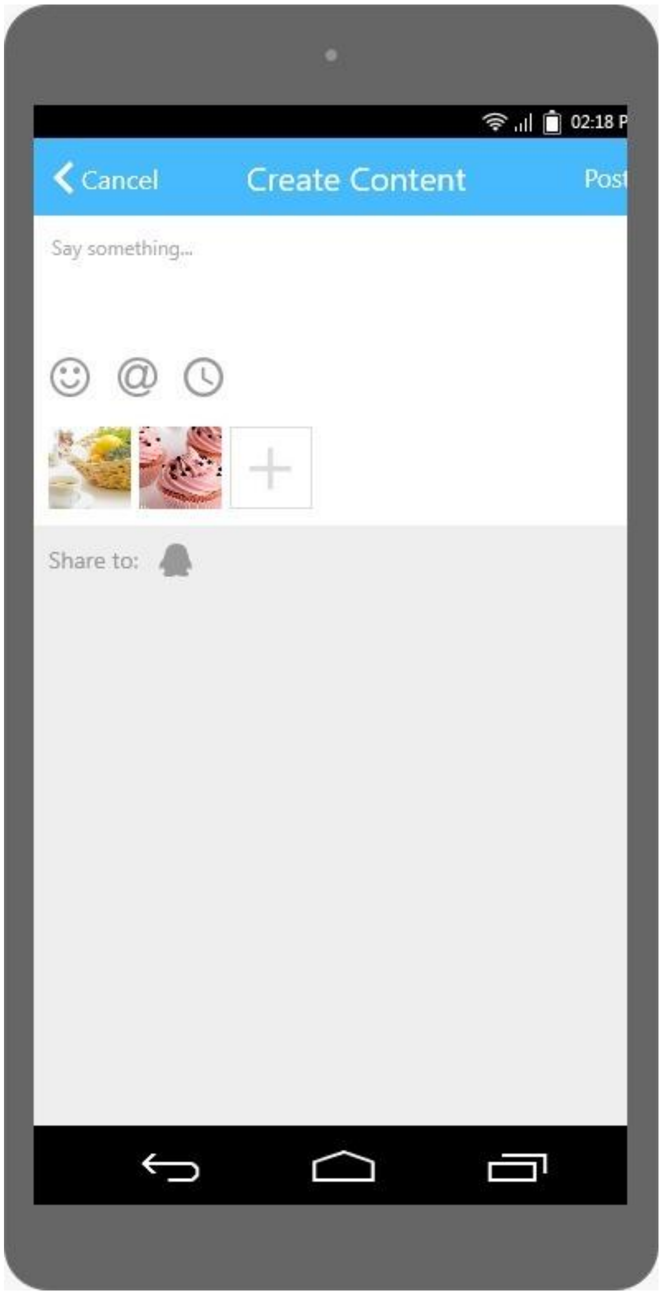


Figure 17

Figure 17 shows the screen in which the user is writing a post and the user can do the following:

1. Upload an image by hitting the + button (1).
2. Tag someone by hitting the @ button (2).
3. Add an emoji by touching the emoji button (3).

4. Add an activity by touching the clock icon (4).
5. Publishing the content by hitting the Post button (5).
6. Cancel what he has done and go back to explore screen by hitting Cancel (6).

7. Photos Screen

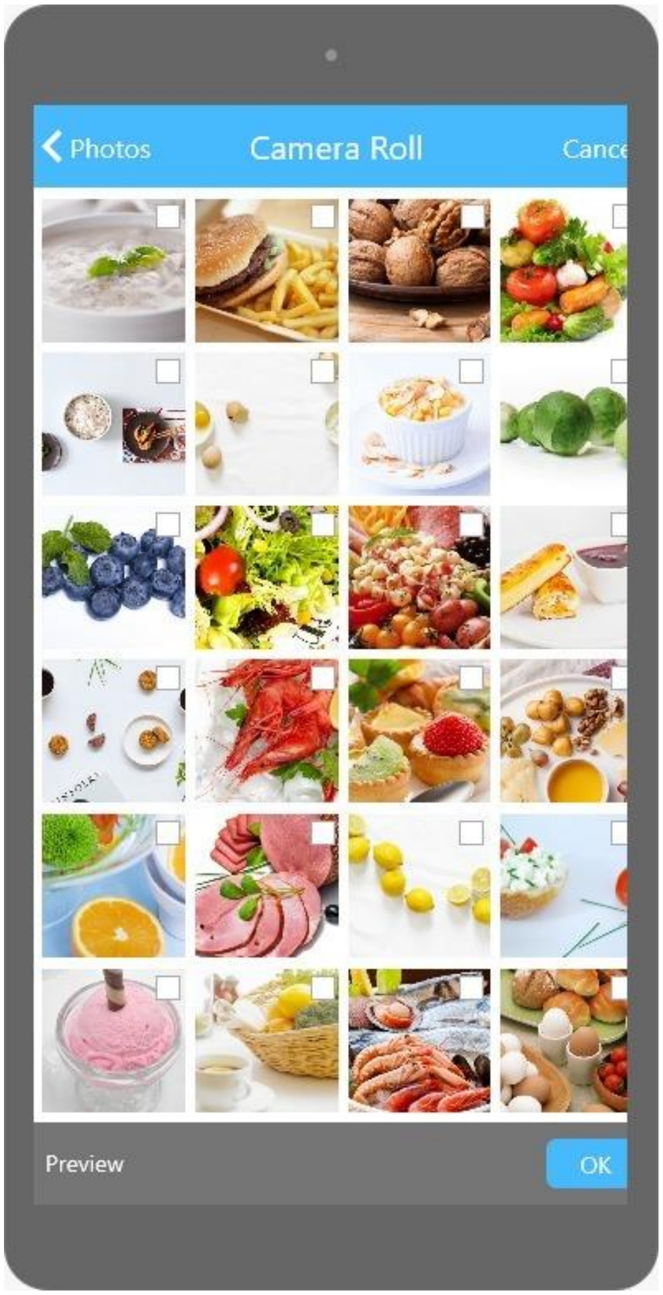


Figure 18

This screen in Figure 18 is displayed when the user wants to upload a photo. In this screen the user can do the following:

1. Preview a photo he selected by clicking the Preview button (1).
2. Cancel the uploading process by touching the Cancel button (2).
3. Upload a photo using the OK button (3).

8. Photos Preview Screen

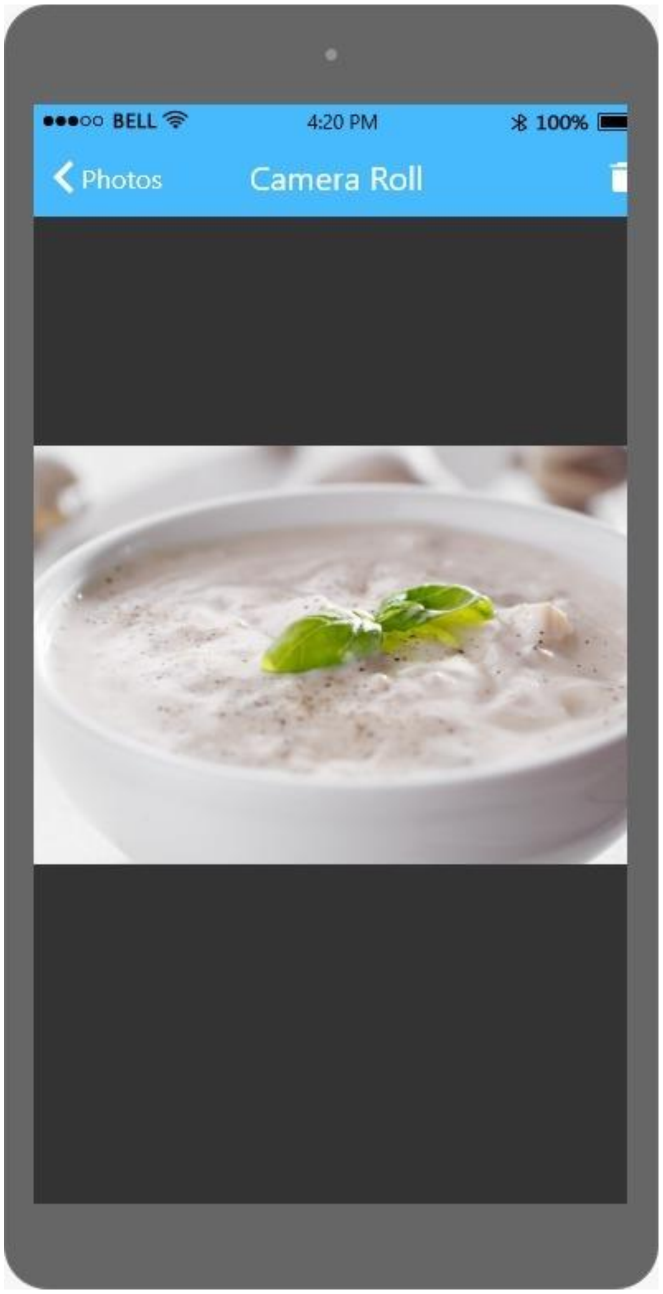


Figure 19

This is the preview screen in which the user can see the photo he is going to upload.
The screen displays the following:

1. The photos button that allows the user to back to the photos screen (1).
2. The delete icon that allows the user to prevent this photo from being uploaded (2).

9. Bots Screen

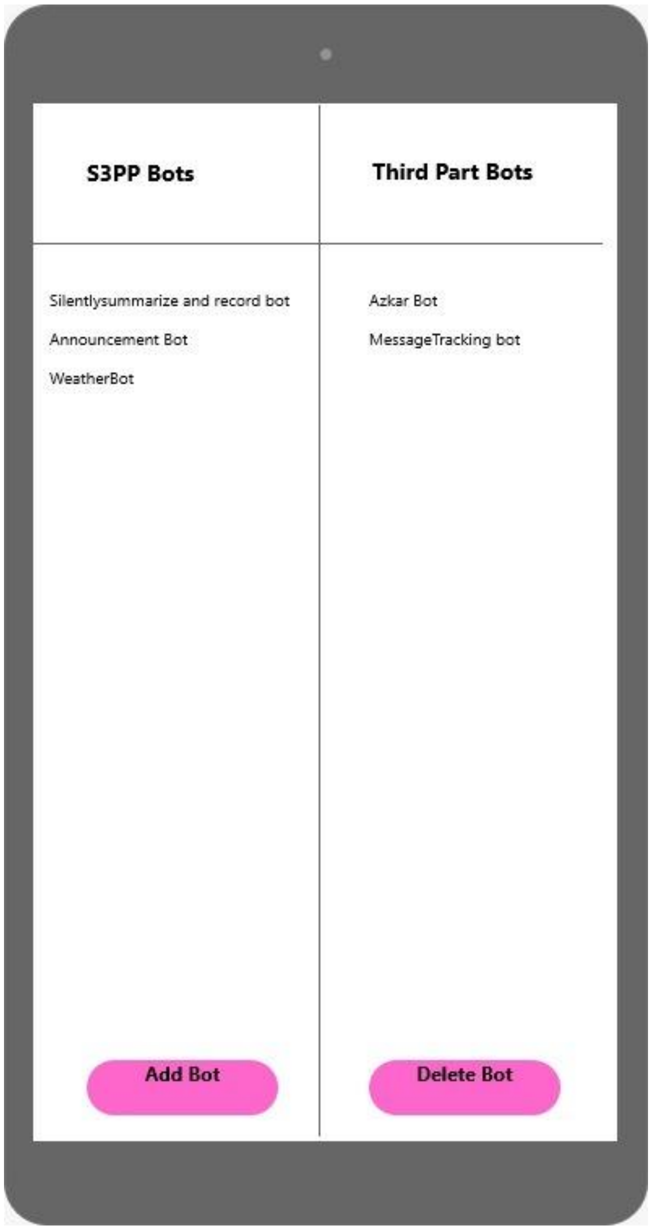


Figure 20

In this screen the user can either use a bot from the market of from a 3rd party vendor. He can also do the following:

1. Add a bot using the Add Bot button (1).
2. Delete a bot using Delete Bot button (2).

5.3 Screen Objects and Actions

The following table describes the objects in each screen of the UI and the action it does.

Welcome screen	1	Button	The object is responsible for submitting the ID and password of the user to log in
	2	HyperLink	The object is responsible for directing the user who wants to create an account for the sign-up page.
Explore Screen	1	Button	This object is responsible for liking a post
	2	Button	This object is responsible for adding a comment on the post
	3	Button	This object is responsible for sharing a post
	4	Button	This object is responsible for adding new content
	5	Button	This object is designed to allow the user to back one step
	6	Button	This object is designed to allow the user to enter the chatting screen
Posting Screen	1	Button	This object is responsible for writing a blog
	2	Button	This object is responsible for uploading photos

	3	Button	This object is responsible for directing the user to the Explore Screen
	4	Button	This object is responsible for closing the prompt
Chatting Screen	1	Text Box	This object is responsible for taking the text input from the user to be sent in the chat
	2	Button	This object is responsible for recording a voice note to be sent in the chat
	3	Button	This object is designed to view the emoji panel to help the user to choose one
	4	Button	This object is responsible for uploading photos or videos to the chat box
Identity management screen	1	Button	This object is responsible for showing the options of managing the identities
	2	Button	This object is responsible for adding new friends in a specific ring
	3	Button	This object is responsible for deleting a friend from a specific ring
	4	Button	This object is designed to allow the user to choose an identity to

			continue using the app with
Publishing Content	1	Button	This object is responsible for uploading photos
	2	Button	This object is designed to allow users to tag friends
	3	Button	This object is designed to view the emoji panel to help the user to choose one
	4	Button	This object is designed to allow users express an activity they are doing
	5	Button	This object is responsible for publishing the content
	6	Button	This object is responsible for canceling the post from being uploaded
Photos Screen	1	Button	The object is responsible for previewing photos
	2	Button	This object is responsible for closing the photos browser
	3	Button	This object is responsible for uploading the photos picked
Preview Screen	1	Button	This button allows the user to back one step to the photo screen

	2	Button	This button deletes the photo from the preview page
Bots Screen	1	Button	This object is responsible for adding a new bot
	2	Button	This object is designed to delete an existing bot

6. Appendices

Figure 21 shows a sequence diagram for the interactions when a user signs-up, logs-in, or adds an identity.

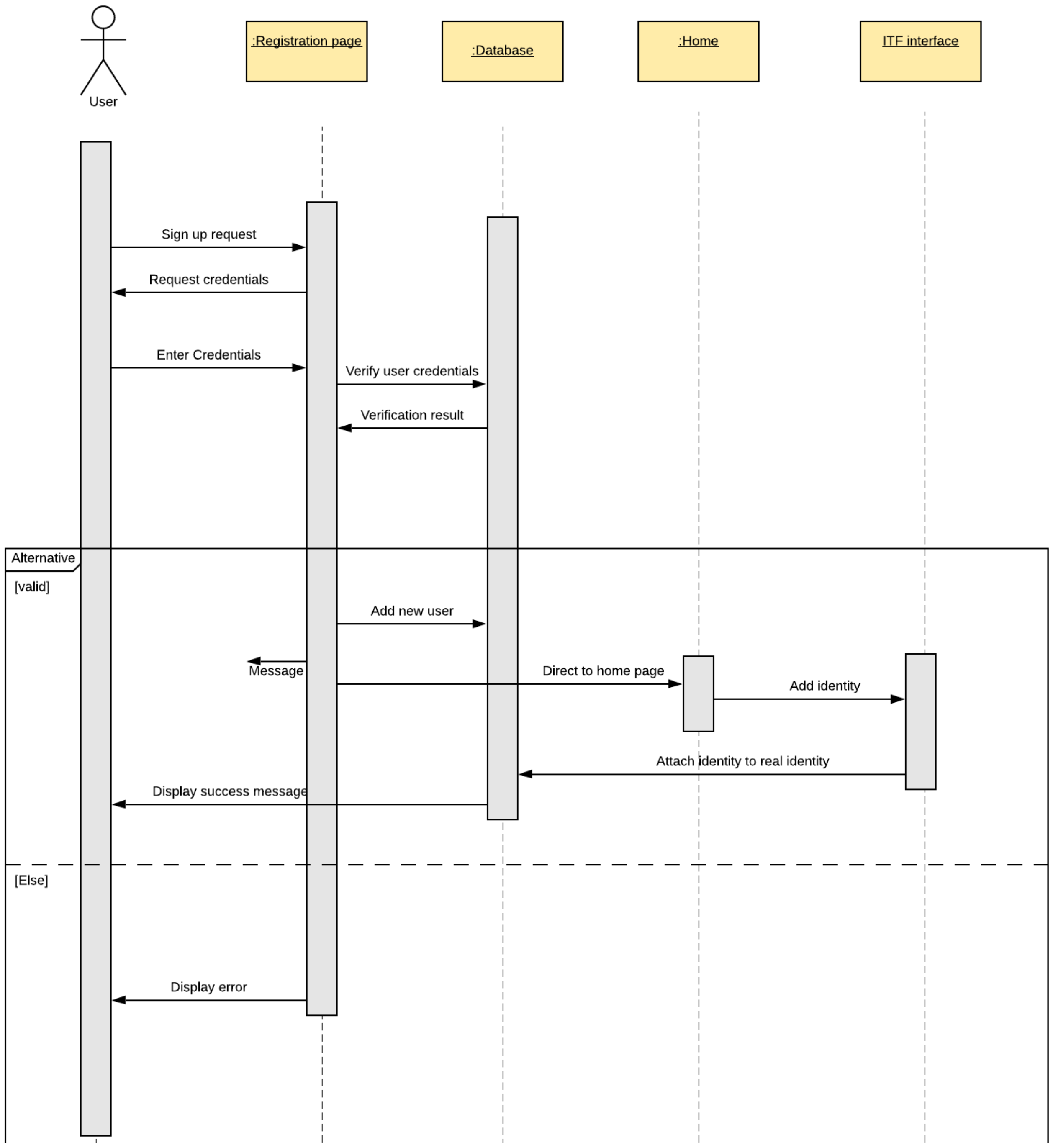


Figure 21

Figure 22 shows another sequence diagram for the interactions when the user selects an existing bot to use or a new one to buy.

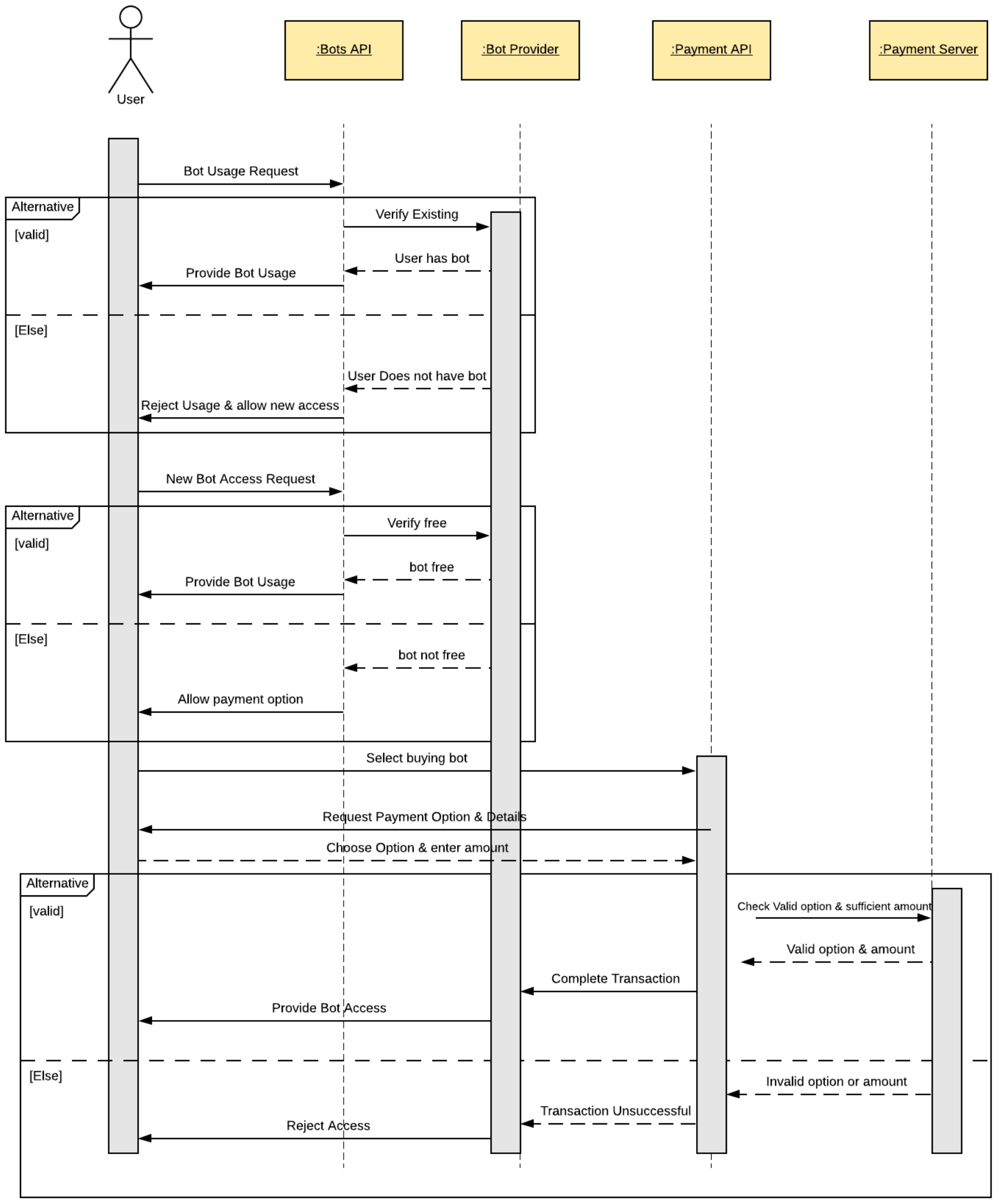

User
:Bots API
:Bot Provider
:Payment API
:Payment Server
Bot Usage Request
Alternative
[valid]
Verify Existing
User has bot
Provide Bot Usage
[Else]
User Does not have bot
Reject Usage & allow new access
New Bot Access Request
Alternative
[valid]
Verify free
bot free
Provide Bot Usage
[Else]
bot not free
Allow payment option
Select buying bot
Request Payment Option & Details
Choose Option & enter amount
Alternative
[valid]
Check Valid option & sufficient amount
Valid option & amount
Complete Transaction
Provide Bot Access
[Else]
Invalid option or amount
Transaction Unsuccessful
Reject Access

Figure 22