Texas A&M University

CSCE 431 - Software Engineering

**FileSwipe**

**Software Requirements Specifications**

John Brock

Irma Lam

Ryan Oji

Clarissa Soon

# Table of Contents

[1. Introduction 4](#_Toc317489575)

[1.1. Purpose 4](#_Toc317489576)

[1.2. Scope 4](#_Toc317489577)

[1.3. Definitions, acronyms, and abbreviations 4](#_Toc317489578)

[1.4. References 4](#_Toc317489579)

[1.5. Overviewk 4](#_Toc317489580)

[2. Overall Description 5](#_Toc317489581)

[2.1. Product perspective 5](#_Toc317489582)

[2.1.1. System interfaces 5](#_Toc317489583)

[2.1.2. User interfaces 5](#_Toc317489584)

[2.1.3. Hardware interfaces 6](#_Toc317489585)

[2.1.4. Software interfaces 6](#_Toc317489586)

[2.1.5. Communication interfaces 6](#_Toc317489587)

[2.1.6. Memory constraints 7](#_Toc317489588)

[2.1.7. Operations 7](#_Toc317489589)

[2.1.8. Site adaptation requirements 8](#_Toc317489590)

[2.2. Product functions 8](#_Toc317489591)

[2.3. User Characteristics 9](#_Toc317489592)

[2.4. Constraints 9](#_Toc317489593)

[2.5. Assumptions and dependencies 9](#_Toc317489594)

[2.6. Apportioning of requirements 9](#_Toc317489595)

[3. Specific requirements 10](#_Toc317489596)

[3.1. External interfaces 10](#_Toc317489597)

[3.2. System Features 11](#_Toc317489598)

[3.3. Performance requirements 14](#_Toc317489599)

[3.4. Design constraints 14](#_Toc317489601)

[3.5. Software system attributes 15](#_Toc317489609)

[3.5.1. Reliability 15](#_Toc317489611)

[3.5.2. Availability 15](#_Toc317489612)

[3.5.3. Security 15](#_Toc317489613)

[3.5.4. Maintainability 16](#_Toc317489614)

[3.5.5. Portability 17](#_Toc317489615)

[4. Index 18](#_Toc317489616)

# Introduction

## Purpose

This document provides a description of the functions and specifications of the FileSwipe software application. FileSwipe targets a large audience, as it will be useful to anyone with a smartphone and a need to share files on it.

## Scope

FileSwipe is designed to be a smartphone based application which allows users to “swipe” and transfer files from one phone to another phone. The user will select the file that they wish to share with another user and they will make a finger gesture to “send” the file to the intended destination.  The phone app will either be able to store the files locally on the phone itself, or it will display a list of the files stored on the web server.  The web server will be used as an intermediate destination between the phones which will handle file transfer, storage, and destination mapping issues.  The goal is to build the project in a manner that is independent of the phone’s OS thus providing a more robust solution to a larger user base.

## Definitions, acronyms, and abbreviations

|  |  |
| --- | --- |
| **Term** | **Definition** |
| FTP | File Transfer Protocol |
| Wi-Fi | Wireless local area network connection – 802.11E |
| API | Application Programming Interface |
| OS | Operating System |
| Framework | Guideline or general structure to ensure compatibility and portability |

## References

IEEE Std. 830-1998: IEE Recommended Practice for Software Requirements Specifications.

## Overview

The remainder of this document is divided into 2 sections; the first explains the factors that contribute to the product design and requirements and provides users a general understanding of how FileSwipe works. The final section provides detailed information on all of the software requirements and actions in full for the software developers’ assistance

# **Overall Description**

## **Product perspective**

FileSwipe is a smartphone based project relying on a web server for file transfer and file storage. The solution will be independent and require little interaction with external resources. The smartphone app will be developed using PhoneGap which is a framework that allows the end product to be compiled for different phones’ operating systems. PhoneGap performs the compile by porting the given functions to each specific phone OS by exploiting API’s in each OS.

### System interfaces

The primary system interfaces are between the smartphone and the web server and vice-a-versa. The smartphone application will send a request to receive a list of the current files stored on the web server to display in addition to the files located on the phone. The web server will respond with a list of files that are associated with a user’s account in order for the smartphone app to display the list. When a file is shared with a user, the web server will push a notification to the destination smartphone. However, the primary operation of the FileSwipe system includes the user’s smartphone and the web server that communicates info regarding the user’s account.

### User interfaces

The user will access the UI of FileSwipe through the smartphone – it will be a web-based application that appears to be a native phone app for each of the phones. The user will open the app and a list or grid of icons representing files will appear in the middle of the screen. A navigation/settings menu will be placed at the bottom of the screen. An interactive box will be placed at the top of screen between the very top and top of the file display. This box / area will serve as the destination for users to “swipe” (drag and drop) file icons into in order to transfer the file to another user. Once the file is “swiped” to this area, a destination drop box or similar menu will be displayed to allow the user to select the destination for the file.



**Drag Here to Share**

Destinations  
Sharing File: \_\_\_\_\_\_\_\_

Destination 1

Destination 2

Destination 3

Add SHARE Cancel

New Share Save Options

*If time permits:* The user will be able to access the system and their account through the web utility. The display will be list based for files and there will be basic options to allow the user to perform similar operations as compared with the smartphone app. Registration of accounts may be easier if we have a web utility since creating web pages is more flexible than creating more smartphone app options.

### Hardware interfaces

The smartphone serves as the means by which the user accesses the FileSwipe solution. The FileSwipe smartphone app runs on the phone and provides the UI for the user. The touchscreen interface on the phone will provide the user with the ability to select files for dragging and sharing and also all selection and navigational tasks. Each smartphone operating system that we will be developing our application for is able to accept textual input through an on-screen, virtual keyboard that will be used to type / select the intended recipients and also search for files, if needed. The wireless hardware built into the phones will allow for connection via Wi-Fi or cellular. The files themselves will be stored on the physical memory (flash) of the phone (if possible) and the physical drives in the remote web servers. The web server itself is comprised of unknown hardware but provides the computing power to handle numerous simultaneous requests. Additionally, the server will have a network interface card for Internet connectivity.

### Software interfaces

The software interfaces are centered around the smartphone and the web server. The smartphone app provides the UI and thus the ability for the user to control the functionality of the FileSwipe system. The software running on the smartphone is specific for the phone OS but was generated from the general solution using the PhoneGap translator. The APIs of each specific operating system can be leveraged to provide access to most features on the phone; PhoneGap seamlessly exploits access to these APIs to ensure the general apps can use specific phone features.

The software on the server side shall be specifically written to handle file/data transfers, processing of the data, and the saving of the files, which are in-turn associated with a particular user’s account to ensure access to the files at a later time is possible. The smartphone app will request a listing of the files on the server and then receive information regarding the users account. The ability to “query” the server for stored files on the server and will allow the user to manipulate and share the files on the server through the smartphone’s UI. Additionally, the files stored locally on the phones will be shared via the web server so a copy of the file is stored on the server for redundancy and to protect against any potential transmission errors. Communication between the smartphone and the server will not occur through an open port but rather through the passing and processing of messages and requests. The software from the smartphone will accept and send messages / data from the web server – files and notifications.

### Communication interfaces

The communication between the smartphone and the web server will be based on basic wireless protocols involving basic HTTP requests. File transfer will be accomplished using FTP or other similar means if FTP is not available via smartphone. Get and Post messages will be used to query / “communicate” with the server from the smartphone app. The general app will be written using HTML with a combination of other web scripting languages and thus the basic HTTP requests will be much easier to send and process. The server will be written in a language that allows for easy web access and thus the complexity of communication is also decreased. The phone OS will manage the network connection automatically and thus our app will be able to use either the Wi-Fi or the cellular data connection for use while running our app.

Since we are relying on secure and user-independent file sharing options, we must be able to authenticate a user when sending and receiving particular files and / or requests. Header information in the leading packet will enable the server to associate the data that will follow with the correct account. Another way verification / authentication may be accomplished is by leveraging the OAuth protocol since there is good security provided and simple implementation. Secure file transfer will also be addressed using a securing protocol or “virtual” transfer only on the server side. File permissions and general permissions will be managed on the server and the smartphone app will require user credentials in order for file access and / or file sharing.

### Memory constraints

Memory constraints would have to be that of either the physical device, be it an iPhone, Android, Blackberry, or etc. or a server through which the device can view the document without having to store it to a local memory card. Because the primary use of the program is the transfer of documents, pictures, and other low data items, the only constraint that memory would impose would be the transfer of video clips and other data heavy media.

### Operations

The various modes of operations in the user organization (e.g., user-initiated operations):

* Accessing application main screen.
* File management.
* Communication with server.
* Communication with other devices.

Periods of interactive operations and periods of unattended operations:

* Interactive operations:
  + FileSwipe account creation.
  + User name/password retrieval.
  + Sending files from the local device or a dedicated server to another user with the FileSwipe application installed and running.
  + Adding descriptions and tags to files that are being sent.
  + Sharing files with social media sites such as Facebook, Twitter, etc.
  + Dragging files by providing tactile input to prompt the generation of destinations for the file to be sent.
  + Adding destinations (the accounts of other users) for documents and other items to be sent to as a function of this application.
  + Accepting items sent through the FileSwipe application from another device.
  + Rejecting items sent through the FileSwipe application from another device.
  + Blocking communications with specific users of the FileSwipe application.
* Unattended operations:
  + User authentication for application and data access.
  + Accepting push notifications while the application is not running.
  + Automatically rejecting content from users who have been blocked.
  + Automatically updating the application.
* Data processing and support functions:
  + Adding items to an application dedicated server from a device.
  + Adding files to the handheld device from an application dedicated server.
  + Deleting items from an application dedicated server.
  + Backup and recovery options:
    - Option for user name/password retrieval.
    - Option to sync and refresh to update data represented on the device.
    - Option to sync files on device to the application server.
    - Option to set up sync frequency.

### Site adaptation requirements

Requirements for the proper functioning of the FileSwipe application are:

* Possession of a smartphone.
* Access to an Internet connection as provided by the user’s mobile carrier data plan or through a Wi-Fi connection.

## Product functions

The following are our application three main functions:

1) File Sharing. FileSwipe allows file sharing between users. The user drags the desire file, which could be a PDF, a document, or an image, from their account to a specified area in the screen in order to share it with another user.

2) File Storage. Each user has an account where they can add files from their phone and delete them as well. The user has access to their files from any mobile device using our application by signing in with their account information.

3) File Viewing. Users have the ability to view the files on their account.

## **User Characteristics**

The audience for our application is any individual who owns a touchscreen smartphone with access to a Wi-Fi connection or a data plan. The user should have basic knowledge of smartphone usage and browsing experience.

## Constraints

The following constraints could be encountered in the development of this application:

Hardware limitations - our application will reside on the smartphone hardware compatible with their operating system.

Interfaces to other applications - other than system level applications, such as network, operating systems, database system, and web server communication. There will not be interfaces with other applications.

Audit functions - our application will create and store log files which will track the application’s activity.

Higher-order language requirements - there are no specific high-level language requirements.

Reliability requirements - our application must be available 24 hours per day.

Regulatory policies - there are no regulatory policies since our application is not going to be available to the public from an application store.

Safety and security considerations - securing user’s information, getting permissions to files stored on the user’s phone and permissions to “share” files between accounts.

## Assumptions and dependencies

No specific assumptions or dependencies.

## Apportioning of requirements

Requirements that will be delayed until future versions of the system:

* Sharing files with multiple users at the same time.
* Storage of other type of files, such as music and videos.
* Implementation of a web application.
* Saving files directly from the phone’s web browser to our application.

# Specific requirements

## **External interfaces**

**3.1.1. User interfaces**

The user shall be able to load the FileSwipe application from within an applicable smartphone such as are within the iPhone and Android brand of smart phones.

**3.1.2. Hardware interfaces**

1. The FileSwipe application shall support the logging and authentication of users.
2. The initial window of the FileSwipe application shall contain two text fields, one for a user name and the other for a password, as well as a button labeled “Log in”.
3. The password text field shall be a field that does not display what the user types but display the length of the password as denoted by a special character for each character contained within the password.
4. When a user presses the “Log in” button, the FileSwipe application shall send a request to the application web server to log in the user.
5. The touch screen of the phone shall provide a method for the user to select files as well as to drag and drop them in the interactive box (see f) for sharing.
6. The touch screen of the smart phone shall provide the primary method for navigating the FileSwipe application.
7. The hardware within the smart phone shall allow network connection to either Wi-Fi or cellular data.
8. The files used in the FileSwipe application shall be stored to the physical memory of the local device or the physical memory of drives used by the web server.
9. The web server shall have the computing power to handle multiple requests and also contain a wireless card for network connectivity.

**3.1.3. Software interfaces**

1. If the logging of a user is unsuccessful due to a lack of a network connection, the FileSwipe application shall display an error message reporting a lack of network connection as the cause.
2. If the logging of a user is unsuccessful due to invalid credentials, the FileSwipe application shall display an error message reporting that an invalid username and password pair have been entered.
3. The FileSwipe shall always display the username of the user who is currently logged in to the application.
4. The FileSwipe application shall display icons representing files in the middle of the screen.
5. The FileSwipe application shall display a menu beneath the icons (see d).
6. The File Swipe application shall display an interactive box at the top of the screen. This box shall allow users to swipe files from the icons area to other users of the application as specified by the user.
7. Pending files to be received, should there be any, shall always be displayed until the user indicates otherwise.
8. A method for adding users to whom files can be sent and received from shall be provided by the FileSwipe application.
9. A method for rejecting/blocking requests from other users shall be provided by the FileSwipe application.

**3.1.4. Communication interfaces**

1. The data stored on the local device shall be able to sync with the data stored to the users account on the FileSwipe web server.
2. The FileSwipe application shall communicate with the web server in order to send files to other users at the request of the user.
3. The FileSwipe application shall communicate with the web server in order to determine if there are any files to be received by the user.
4. The phone operating system shall manage the network connection. This communication shall take place at a frequency that the user dictates.
5. The range of the FileSwipe application shall be any other applicable smart phone currently connected to a network.
6. The FileSwipe web server shall manage file permissions and general permissions.
7. The local device shall manage authenticating user log in credentials.

## System Features

This section documents the functional requirements of the FileSwipe application and its features to help in the development and testing progress of the system.

**3.2.1. System Feature 1 – Account Creation**

**3.2.1.1. Description**

The user can create an account that can be uniquely identified through their mobile phone’s telephone number. Accounts will be registered through a web interface but if time permits then the ability will be incorporated into the actual smartphone app.

**3.2.1.2. Stimulus / Response Sequence**

If a user does not already have an existing account, user has to select a “Create Account” option which will ask for specific information such as name, phone number, and password to create the account. Upon submission of the information, an account will be created which the users can then use to manage file sharing options.

**3.2.1.3. Functional Requirements**

Requirement 1:

User’s mobile phone must be connected to Wi-Fi or 3G.

Requirement 2:

Valid mobile phone number

Requirement 3:

Confirmed password / optional security PIN

**3.2.2. System Feature 2 – Adding Contacts**

**3.2.2.1. Description**

The user is able to add other users as their contacts. Files can be shared between contacts.

**3.2.2.2. Stimulus / Response Sequence**

After dragging a file to the designated sharing area, a list of contacts will appear. If the intended recipient is not among the user’s current contacts, the user can add someone to their contacts using the "Add" option. It will ask for the other party's phone number, as well as information such as First and Last Name to keep track of the contacts. Once this information has been submitted, it will send the request to the number provided and contacts will only be established when the request has been accepted by the other party.

**3.2.2.3. Functional Requirements**

Requirement 1:

User’s mobile phone must be connected to Wi-Fi or 3G.

Requirement 2:

The other user must have FileSwipe installed in order to receive the request.

**3.2.3. System Feature 3 – Uploading Files**

**3.2.3.1. Description**

Once users has logged into an account, they are then able to upload a particular file to their account (on web server), which can later be shared to another user if desired.

**3.2.3.2. Stimulus / Response Sequence**

In the main UI menu after logging in, all files associated with the user’s account will be displayed and there will be a “Add” option to add a file to the account on the server. The user can then browse the directory where the file is located, select the file and upload it.

**3.2.3.3. Functional Requirements**

Requirement 1:

User’s mobile phone must be connected to Wi-Fi or 3G.

Requirement 2:

Only the following file formats can be uploaded:

* PDF
* DOC
* TXT
* JPG

Requirement 3:

The user’s account size is limited to 100mb. Total files on this account cannot exceed this size.

**3.2.4. System Feature 4 – Sharing Files**

**3.2.4.1. Description**  
Files which have been uploaded can then be shared with other contacts.

**3.2.4.2. Stimulus / Response Sequence**

In the main menu with the list of files, the user can simply select this file by touching the icon, and swiping it to the designated area that says “Drag here to share”.  It will then prompt the user and ask which contact to send the file to.

Once the sender requests a file to be sent, the receiver will receive an acceptance request where he or she can choose to accept or reject the file.

File is successfully shared when the receiver accepts the file. The server will duplicate this file into the receiver’s account.

**3.2.4.3. Functional Requirements**

Requirement 1:

User’s mobile phone must be connected to Wi-Fi or 3G.

Requirement 2:

Both the sender and receiver must have FileSwipe opened to be able to send and receive the File request.

**3.2.5. System Feature 5 – Deleting Files**

**3.2.5.1. Description**

In addition to sharing files, the user will also be able to view or delete files that are in the user’s account on the mobile device.

**3.2.5.2. Stimulus / Response Sequence**

Also in the files list next to the “Drag here to share” box, there will be a trash can image icon. If a file is dragged to this icon, a message will prompt the user for confirmation in deleting the specified file. Once confirmation is received, the file will be removed and deleted from the user’s account.

**3.2.5.3. Functional Requirements**

Requirement 1:

User’s mobile phone must be connected to Wi-Fi or 3G.

Requirement 2:

To view the file, phone must support the file format.

## Performance requirements

Our application will connect to a web server, which provides response times that will not take longer than 1 second, unless the server is down or is experiencing performance issues. Information handled from the user will be in the form of text and PDF files, and images, as described in previous sections. The web server will be in charge of performing storage, deletion, and transfer operations according to the user’s requests and actions. We will give each user a fixed amount of storage space in the server.

Checking the fact that the system must perform as what every user expects, in every action-response of the system, delays should not be take more than 1 second if there is a reliable Internet connection, and the server is working properly.

In case of transferring files, if there is an error during the transfer, an error message will be displayed. File transfer delay is less than 2 seconds once the receiver has accepted the transfer. In case of opening databases, writing, reading from the database, there are no delays, and the operation is performed in less than 2 seconds.

Also when connecting to the server, the delay is based on the distance of the 2 systems and the configuration between them, so there is a high probability that there will be a successful connection in less than 20 seconds.

## Design constraints

FileSwipe is intended to be a platform independent mobile application that will work for iPhone, Android, Windows Phone and Blackberry. It will be written in HTML5 and Javascript through the PhoneGap build which will allow the app to be deployed across different platforms. Therefore, anyone who wishes to work on further development of FileSwipe has to know and have these technologies

Other constraints on FileSwipe include the size of the screen for each phone. Since every phone has a different screen size, FileSwipe has to be designed in such a way that it would look right and easy to browse on different screen sizes. Files which can be shared and viewed on FileSwipe is also limited to those which the different platforms support. The size of the web server will also put a constraint on how much file space is awarded to each account, and how many accounts which can be created.



## Software system attributes

FileSwipe will be a reliable file sharing solution that also provides each user with dependable file storage/backup and thus the attributes outlined below are baselines for the final solution.  The main components, smartphone app and the web-server, will be working in tandem to share, store, and manipulate files thus both components must be functional for the system to operate correctly.  The measures outlined below are based on averages and consistent operation.



### Reliability

Probability of failure-free software operation for a period of 1 week, 7 days, in a normal, “everyday” environment will be 80%.

80% was chosen because reliability is important but due to time constraints and the fact that our solution is in a very early stage, a higher percentage would be difficult to achieve. The reliability is the product of the reliability of each component and as a result of new development; we are unable to promise a reliability number exceeding 80%.

### 3.5.2 Availability

Percentage of time the system will be “available” / accessible to the end-user should be 90%. Availability = (Uptime divided by (Uptime+Downtime)) and thus we predict the system will be available to the users (.9) = 151.2 hrs / 168 hrs -- 151.2 hours per week implying a downtime (scheduled and unscheduled) of 16.8 hours. The downtime can be a result of malfunctions, network issues, repairs, maintenance, and any other factor that prevents the users from accessing / using the FileSwipe system.

Our solution will be hosted on a 3rd party site (Heroku or Google Code) and their availability will affect our availability - considering the infancy of our solution we feel this is not a significant factor at this stage of development and implementation.

### Security

FileSwipe will be processing, storing, and sharing files for and between users thus security is a significant issue. Files are sensitive to the owner and thus we must ensure access to the files is restricted to the correct owner(s) and that our system is protected from any outside attacks or attempts to access information without authorization.

The smartphone app will connect to the web server through the user’s account and the account shall be available if and only if the phone number and the password / access codes are verified. Physical access to the phone tied to a particular number is a significant security measure due to the fact that physical access is one of the most difficult security measures to overcome. However, there are still ways around this requirement such as spoofing and sniffing, but we will work to ensure that access to files is very secure due to the sensitive nature of files.

The goal is to have 2 layers of security to ensure access to accounts is protected - 1) access to the phone (physical) and 2) password/access code. Additionally, we may look into implementing an additional measure of security by allowing the users to opt into a method that sends a random verification number to the physical phone number upon attempting to log into the account to ensure the user attempting to access the account does in fact have the actual smartphone.

File security in the database / on the server is also a large concern because hosting our app on a 3rd party solution reduces our ability to customize security / permissions. However, this is also a benefit since these solutions already have proven security protocols in place. Access and security will be closely monitored and logged.

File permissions and accessibility will be ensured by having specific permissions on each file tied with the account each file is associated with. Sharing introduces a complication since multiple users will be sharing the file but there will only be one copy of the file on the server. An additional complexity is if we offer the users to share & collaborate or share a copy; if the copy option is selected, a copy of the file is shared, but collaborate will require the same file have multiple user permission options enabled. Time is a factor and this may be beyond our scope at this point of development.

### Maintainability

The four types of maintenance include adaptive, perfective, corrective, and preventive and each will be relatively easy considering our code will be hosted through a shared online repository and hosted on a 3rd party cloud server that allows easy deployments of changes. The adaptive maintenance will address any changes in software environment and the perfective portion will be focused around tuning and adding features to satisfy specific requirements or to address requirements that were not met during the development stage. The final two maintenance categories, corrective and preventive are important because they have direct impacts on reliability and availability but are often sacrificed due to business needs and/or demands for additional features.

During development, we will focus on modular and easily testable code to ensure maintenance difficulty is not increased. Also, there shall be checks and verifications required before new code is pushed to our production environment to prevent unintended instability or issues for the end-users. Continuous integration will also help ensure ongoing development does not impact the overall functionality and reliability of the entire system. Maintenance can be difficult but Agile approaches and test driven development leads to a code base that is easier to maintain in the long run.

### Portability

FileSwipe, as discussed before, is based on two major components, the smartphone app and the web server. The smartphone app shall be developed using PhoneGap to allow portability between different phone operating systems and different versions of the same operating systems. The PhoneGap framework allows general functionality to be ported to specific phone functions through APIs that are exposed by the popular and main-stream phone systems currently on the market. Developing the app as a web app allows cross platform portability that is not achieved using native smartphone app development approaches.

The web server will be written in a web-compatible language such as Java, Python, or Ruby to ensure browser support and also to allow deployment to different hosting solutions. Each of the cloud hosting services vary slightly in their specific implementation and deployment details but the code and applications will execute in a similar fashion as a result of us being intentional with our coding practices. The primary reason for developing a general web server framework is to ensure our app will scale and a different hosting service could be easily implemented in the event one malfunctions or we need to roll our service to a different platform for an unforeseen circumstance.

Online code collaboration, continuous integration, and shared repositories will be leveraged to ensure our code base is robust and portable between smartphones and web hosting solutions.

# Index

account, 6, 7, 8, 9, 10, 13, 14, 15, 16, 17, 18, 19

Android, 8, 12, 17

API, 4, 6

Blackberry, 8, 17

file sharing, 8, 10, 14, 17

FileSwipe, 1, 4, 6, 7, 9, 10, 12, 13, 14, 16, 17, 18, 19

Framework, 4

FTP, 4, 8

iPhone, 8, 12, 17

OS, 4, 6, 7, 8

PhoneGap, 6, 7, 17, 19

smartphone, 4, 6, 7, 8, 9, 10, 12, 14, 17, 18, 19

web server, 4, 6, 7, 8, 10, 12, 13, 15, 16, 17, 18, 19

Wi-Fi, 4, 7, 8, 10, 12, 14, 15, 16