**Software Requirements Specification**

**for**

**TimeTuck Application**

**Version 1.0**

**Prepared by Adam Greenstein, Philip Scott-Berman,**   
**Weeraphat Amornrattanapong, and Garrison Dickey**

**January 21, 2015**

**Table of Contents**

**Table of Contents**

**1. Introduction**

1.1 Purpose

1.2 Document Conventions

1.3 Intended Audience and Reading Suggestions

1.4 Product Scope

1.5 References

**2. Overall Description**

2.1 Product Perspective

2.2 Product Functions

2.3 User Classes and Characteristics

2.4 Operating Environment

2.5 Design and Implementation Constraints

2.6 User Documentation

2.7 Assumptions and Dependencies

**3. External Interface Requirements**

3.1 Hardware Interfaces

3.2 Software Interfaces

3.3 Communications Interfaces

**4. System Features**

4.1 System Feature 1

4.2 System Feature 2

4.3 System Feature 3

**5. Other Nonfunctional Requirements**

5.1 Performance Requirements

5.2 Safety Requirements

5.3 Security Requirements

5.4 Software Quality Attributes

**6.** **Other Requirements**

Appendix A: Glossary

Appendix B: Analysis Models

Appendix C: To Be Determined List

# **Introduction**

## **Purpose**

The purpose of this document is to give a detailed description of the requirements for the TimeTuck app system. It will illustrate the purpose and complete declaration for the development of the system. It will also explain system constraints, interface and interactions with other external applications. This document is primarily intended to be proposed to a customer for its approval and a reference for developing the first version of the system for the development team. The TimeTuck app system will be designed and developed solely by the development team and will not be receiving feedback from an external client.

## **Document Conventions**

The conventions of this document are very simple. Important information will either be highlighted in bold or in quotes. Additionally, this document is broken up into sections and sub-sections. This helps with the logical organization of this document.

This document refers application specific items using applications specific terms. Therefore, it may be necessary for users to reference the glossary for any unknown terms that they may encounter that are not immediately defined in the document. They should be defined in the glossary.

Anything that is not determined shall be labeled as TBD, which stands for “to be determined.”

Finally, the language of this document shall be specific but intended to be understandable by a wide range of readers with different skill levels and backgrounds. The language is not intended to be only understood by programmers or developers. However, there are certain specific items that may only be understood by computer professionals. If the readers do not understand any particular bit of information and it is not in the glossary, then the user may need to research it on their own to fully understand this document.

## **Intended Audience and Reading Suggestions**

The intended audience of this document shall be software engineers, developers, project managers and testers who may contribute to this application. This document will use both technical as well as broad terms to describe each of the following sections, allowing the document to be well understood by individuals with a wide range of knowledge. The document that follows will be organized into five ‘top-level’ sections in addition to this section. Each section will be further divided into subsections. The ‘top-level’ sections will include, ‘Introduction,’ ‘Overall Description,’ ‘External Interface Requirements,’ ‘System Features,’ ‘Other Non-Functional Requirements,’ and ‘Other Requirements.’

The ‘Introduction’ section of the document will give a broad overview of the document that follows. The ‘Overall Description’ section gives a broad description of the TimeTuck application. The ‘External Interface Requirements’ section gives a description of the interfaces, which will be used for this application. Finally, the ‘System Features,’ ‘Other Non-Functional Requirements,’ and ‘Other Requirements’ sections each describe features and requirements that the application will adhere to.

Each section is self-contained. However, this document is best in order from start to finish as certain concepts may build upon one another.

* 1. **Product Scope**

TimeTuck is an iOS mobile application which allows users to create "time capsules" that will be tucked away until a time that they specify in the future. These time capsules consist of images and text that represent an "event."

The way TimeTuck works is simple, a user creates and "event" within the app for themselves and they can share it with friends. Whoever has access to the event can become a contributor. The contributor can take pictures, videos, or write blog posts and add it too the event. No other user can see what has been contributed to an event at the time the event is being created. Once an event is completed, the user who created the event will "tuck it away" until a specified time in the future. While the event is tucked away no trace of the event will exist on the users mobile devices. Once the date in the future arrives, the event will reappear on the contributors devices. It is then that they can see all the media that has been contributed to the event.

The TimeTuck apps goal is to allow users to reminisce about the past in a social setting. This will create an exciting application where users will excitedly await content that they have created in the past to reappear on their device, some of which they may have forgotten about.

While the functional goal of this application are simple, there are also many ways that the application can generate revenue. Generating revenue will not be a primary concern at the early stages of the lifespan, however, it will certainly be kept in mind for the future.

This application, while being built primarily on Apple Computer’s iOS platform at the start, will likely need to be refactored and other platforms in the future. The design and implementation will need to keep this in mind to ensure maximum portability and agility.

## **References**

[1] Apple Inc., *Swift Programming Language*, Apple Inc., 2014.

[2] Apple Inc., *Using Swift with Cocoa and Objective-C*, Apple Inc. 2014

[3] PayPal Developer website, <https://developer.paypal.com/webapps/developer/docs/>,

2014

[4] Parse Documentation website, <https://parse.com/docs> 2014

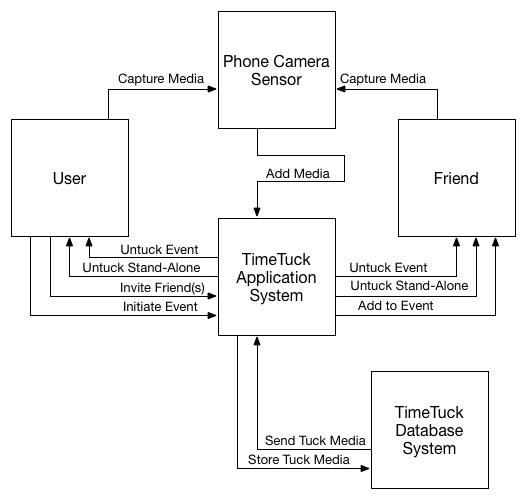
[5] Flask website, <http://flask.pocoo.org/> 2014

# **Overall Description**

## **Product Perspective**

TimeTuck is a new, standalone iOS application that allows users to create time capsules of vents or images which will be sent back to the users at a specified time in the future. Users will register accounts and will have the ability to store media on our servers which will allow the application to function. There will be a database server as well that will store user information and will run scripts that will push media to user's mobile devices that are running the TimeTuck application.

Figure A



## **Product Functions**

The main purpose of the TimeTuck application is to allow users to create time capsules of media and send them to themselves or friends at specified times in the future. This will create a social setting where users can reminisce about events in the past. Unlike other "time-capsule" like applications, this app will require users to actively contribute content to the time-capsule.

Users begin by creating an account or logging into a previously created account. Users can view search for other friends through the application who are using the app and connect with them. Users can then take pictures, videos or write short blog posts that can then be added to an event or become standalone media. Stand alone media is media that is not associated with an event that has been created by a user. Events are created and can either be shared with friends or remain only available on the user who created it's phone. Anybody who has access to the event can contribute content to it. While the event is in the creation stage, no user, including the event's creator, can see what media has been contributed to it. Once the user who created the event is ready, they can "tuck-away" the event until a time in the future.

All contributors to the event will receive the event when it is "untucked." The creator can also specify who else can see the event. Once an event has been "untucked" it will never go away. All users can then download the media to their device at that time.

## **User Classes and Characteristics**

There is one type of user class for the TimeTuck application. This is the main user type. This is anybody who is able to create media and contribute to events. All users can make connections with other users.

## **Operating Environment**

The software will operate on the following environments:

* Apple iPhone 4S or newer
* Apple iPad 2 or newer, cellular models only
* Apple iPad mini or newer, cellular models only
* Apple iOS 8 or newer

In addition the system will use the following third-party components:

* Parse.com web server or Flask with django.
* Media server

## **Design and Implementation Constraints**

While the system is set to be implemented primarily using the Swift programming, other languages and frameworks such as Objective-C, Cocoa, and Python will used in addition. One major design consideration will be the implementation of the backend for the application. It will be desirable for performance sake to have a heavily customizable backend. This will allow us to make requests to the backend and allow processing on the backend side. This will prevent data from being sent back and fourth between server and the application. Using backend such as parse may limit our ability to process data on our server and thus may require redundant or unnecessary data transfers. Using a framework such as django or Flask may be desirable to fit the performance constraint. However, this may significantly increase development time, therefore it will be important to find a balance.

## **User Documentation**

This application will be simple and have minimal screens to lessen the learning curve that users face. Because of this extensive documentation is not necessary. However, there will be the following documentation available to the users:

1. Tutorial in the app - The first time user uses that application there will be pop-ups that highlight features that may not be obvious at first glance. This will include where to go to add vehicles and how to create a ride.
2. Website documentation - While a website component will not be part of the first few increments of this application there is desire to implement this in the future. When this occurs there will be documentation available online. This is not a necessary component at the start as TimeTuck will remain simple, however, as development continues, the application will likely grow in complexity, which will make this website-based documentation more relevant.

## **Assumptions and Dependencies**

A number of factors that may affect the requirements specified in the SRS include:

* The user will have a iOS device.
* The user will have a camera on their device.
* The user will be tech savvy and understand the purpose of the application before use.
* Backend web server will be dependent on Parse.com or which framework is chosen.
* Users are assumed to have standard knowledge on how to use iOS.
* The users will have a device that has an internet connection and the ability to track the user’s location.

## **External Interface Requirements**

## **Hardware Interfaces**

This application will only be supported on Apple’s iOS operating system so it will only be available on the iPhone and iPad. Because this application does require an Internet connection to take full advantage of our back-end systems and web services, the device will have to have access to the Internet. This may limit its use specifically on iPads to those who have cellular data enabled, which may not be the case for all iPads.

The application will communicate with our back-end Parse.com database via a JSON based web service or a custom Flask based web-service. This web service is accessible through a native code based API supplied directly through Parse.com and implemented through the software. This communication will physically take place through the device’s network connection.

Additionally, Parse.com will communicate with external web services through Parse.com’s physical web connection to pull data from external web services that will be cached in the Parse.com database. This will enable the application to only connect with Parse.com and not the external web services, thus minimizing the potential network errors that may occur from accessing other external physical web services.

The above will be the design if Parse.com is the only backend that is chosen. A similar design will still be used if a more customizable framework is chosen. The only difference is that there will be more processing which will occur when data is fetched, that way, less redundant or unnecessary information will be transmitted.

It will be more desirable to handle everything in a custom web-service if development time permits. Flask is the chosen framework as it works with Python and we are very familiar with this language.

## **Software Interfaces**

The software in this application will contain the following components:

1. iOS Application
2. Mongodb database with parse or MySQL with our own framework
3. External Web Services

The iOS Application will utilize the iOS operating system interface to be able to utilize the software and hardware based features available on the device. This will include, but are not limited to, the ability to track the location of the user and the ability to access an Internet connection.

The iOS application will also utilizes iOS’ background services to allow the application to track the location of the user when the screen is off or the application is in the background.

The Mongodb database will be stored on Parse.com. Parse.com has an API interface that can allow the application to communicate with the database naively.

If we create our own custom framework, then we will be using MySQL. Python has a handy module that allows us to connect to our MySQL database very easily.

## **Communications Interfaces**

Because this application will be run on mobile devices, there will be an important emphasis on the communication interfaces. This is due to the fact that almost all of the data gathered with the application will have to be transmitted back to the Parse.com or any other framework to store on our backend database.

The information will be transmitted over HTTP to Parse.com’s databases or our custom database. It will be ideal if this data can be transmitted securely using HTTPS. This will need to be determined (TBD) if Parse.com has support for this feature or if we will need to use our own system. This will ensure that all the data that is being transmitted is being transmitted securely. This is important because some of the data that is being transmitted may be sensitive, including location information. The location information is considered to be sensitive data.

The application should be able to locate where the users are at least in future versions of the app. This utilizes the devices GPS systems that enable the system to know where the user is. The iOS GPS system is very reliable and fast. In fact, the GPS system is much more reliable then other types of GPS systems because the iOS devices are able to utilize cell towers and Wi-Fi hotspots to accurately find GPS satellites very quickly. These handy features will make this application fast and very reliable.

# **System Features**

The following section will highlight the important features that will be part of the TimeTuck application. The features section is divided into subsections, each describing a particular feature. Each feature section is further divided into subsections including a Description and Priority, Stimulus/Response Sequences, and Functional Requirements.

## **System Feature 1**

4.1.1 Description and Priority

* Users have their own accounts and can connect with other users - Priority Level High

4.1.2 Stimulus/Response Sequences

* Users press the sign up button to create an account. This takes them to the sign up page where they can enter their information and click the submit button. This creates their account.
* Users can press the login button after they update their information. This logs them into the app.

4.1.3 Functional Requirements

REQ-1: Users shall create an account by logging in with Facebook, Twitter, or email with verification.

REQ-2: Registration involves, username, email, password, and phone number.

REQ-3: All users shall register with the system.

REQ-4: There shall be one type of account

REQ-5: Users can connect with other users in the app (friends)

REQ-6: Users can search for other users by phone number, email, or username.

REQ-7: Users can search their phonebook for other users who have the app.

REQ-8: Users send friend requests to other users

REQ-9: Users receive notifications when a friendship request is sent to them.

REQ-10: Users must accept a friendship request

REQ-11: Users can delete friends

## System Feature 2

4.2.1 Description and Priority

* Users will capture media and tuck it away - Priority Level High

4.2.2 Stimulus/Response Sequences

* With the user logged in the app, the app will open in the camera mode ready to capture media.
* User will press the shutter button and confirm if they want to use the media by pressing an arrow on the lower right-hand side.
* The following screen will present a calendar for the user to schedule when the media will untuck.

4.2.3 Functional Requirements

REQ-1: Users take pictures or videos in the application.

REQ-2: Users take pictures or videos in the application.

REQ-3: Pictures and videos are not stored on the device.

REQ-4: Pictures and videos can also be immediately tucked away and not added to an event (stand-alone).

REQ-5: Standalone media only has a capture date.

REQ-6: Each piece of media in the event has the time it was added and the user who added it.

REQ-7: Images or videos can be downloaded once an event has been untucked.

REQ-8: Users can add a caption to the image.

## **System Feature 3**

4.3.1 Description and Priority

* Users can create events and share them with friends

4.3.2 Stimulus/Response Sequences

* User navigates to the home page and click the create event button.
* They can add a name and description to the event
* Users can then invite friends to the event.

4.3.3 Functional Requirements

REQ-1: Any user can create an event.

REQ-2: Events can be created before or after images are taken.

REQ-3: Events can be shared with friends.

REQ-4: Users receive notifications when an event is shared with them.

REQ-5: Each picture and video will only be added to one event.

REQ-6: While an event is being contributed to, no user, including the creator can see what is in it.

REQ-7: User who creates an event is the one who tucks it away.

REQ-8: User who tucks event or stand-alone media away sets a time in the future that the event or stand-alone media will reappear.

REQ-9: User can specify additional friends who the event or stand-alone media will be sent to in addition to the contributors.

REQ-10: When an event is untucked (reappears) all contributors can see the event.

REQ-11: Each event will have the start date and the end date of when it was contributed to.

REQ-12: Users can view their history of untucked events or stand-alone media.

REQ-13: Users who contributed to an event or who the creator specified receive notifications when an event is untucked and is available for them to view.

REQ-14: Users can only view their own history of untucked media

# **Other Nonfunctional Requirements**

## **Performance Requirements**

This application does not have very strict performance requirements. However, it is necessary that this application works well and quickly. We will be uploading and downloading images from a server. Therefore, we should be able to do this quickly and with ease. It is desirable that most of the processing will be done on the server, that way we don not have to make many database calls directly from the app.

Because this is a real time system, that will utilize an active Internet connection, it is important to take the necessary considerations to ensure that the software functions properly all phones. Unfortunately, all phones cannot be expected to have the fastest Internet connection. This application will need to function well on any phone whether or not it is using a 4G connection or a 1G connection.

## **Safety Requirements**

There will be some safety requirements for this application. However, these will probably not be implemented in the first few iterations of the application.

Some of the safety features could be a feature that allows users to flag inappropriate content or content that can be considered harassing or offensive. These will not be necessary at the beginning and will likely be implemented later on in the lifespan.

## **Security Requirements**

There are a number of security and privacy concerns with this application:

1. Images must be stored securely so that they cannot be accessed by unintended people.

To ensure that user information is stored securely we will be encrypting all location information in our databases so that they will not be readable in the unlikely event that our systems are compromised. Location information is sensitive data that many users will be much more comfortable with knowing that this information cannot be made available.

* 1. **Software Quality Attributes**

Quality will be an important concern for users for various attributes. Users will want to ensure that the system will reliably and all images and media should be pushed to a device when it is scheduled to.

1.

# **Other Requirements**

An additional requirement is that there will have to be a program running on our application server that will read through our database and send any push notifications to specific devices that need a push notification sent to them. This will have to not put a heavy load on our database so that users who are using the app will not have any performance hits.

**Appendix A: Glossary**

The following are a list of useful terms that may be important or specific to the TimeTuck app:

1. User - person who has a TimeTuck account
2. Friends - connection between two users
3. Event - an activity that users can contribute to
4. Contributor - a user who contributes to an event.
5. Tuck-away - when a time in the future is set for an event or stand-alone media
6. Un-tuck - stand-alone media or event that reappears
7. Stand-alone media - pictures or videos that are not added to an event, but are independent and immediately tucked away
8. Parse.com - web service that will serve as the backend for this application
9. iOS - Apple's mobile operating system that will support the TimeTuck app
10. API - Application Programming Interface

**Appendix C: To Be Determined List**

1. Determine the type of encryption to use to ensure if images and user data is secure.
2. Determine legal liabilities and a term of agreements for users.