Feasibility Evidence Description (FED)

Populic

Team No.4

Name	First Role	Second Role	Email Address
Chengyu Shen	Project Manager	Operational concept developer	**shenchen@usc.edu**
Shiji Zhou	Prototyper	Software Architect	**shijizho@usc.edu**
Yufei Hong	Feasibility Analyst	Project manager	**yufeihon@usc.edu**
Guanghe Cao	Software Architect	Life Cycle Planner	**caog@usc.edu**
Yang Wei	Operational concept developer	Prototyper	**Wei495@usc.edu**
Lin Xia	Life Cycle Planner	Feasibility Analyst	**xialin@usc.edu**
William Goishi	IIV & V	Quality Focal Point	**wgoishi@usc.edu**

Version History

Date	Author	Version	Changes made	Rationale
10/01/17	Yufei Hong	1.0	• Original template for use with Yufei Hong v1.0	• Initial draft for use with Yufei Hong v1.0
10/11/17	Yufei Hong	1.1	• Modify the Personnel Costs • Modify the Hardware and Software Cost • Modify Risk Assessment • Modify the ROI Analysis • Modify the Benefit Analysis • Modify the LOS Feasibility	• Fix the wrong concepts and data of these parts. v1.1
10/15/17	Yufei Hong	1.2	• Modify the ROI part	• Fix the wrong data of ROI form and fix the ROI graph
11/30/17	Yufei Hong	1.3	• Revise and modify Personnel costs, Level of service feasibility and Process feasibility	• Some of the rationales and concepts are incorrect, has to be changed to correct and reasonable solutions
12/17/17	Yufei Hong	1.4	• Revise and modify ROI	• The unit of ROI is incorrect and has to be fixed

Table of Contents

Table of Tables

Table of Figures

1. Introduction

1.1 Purpose of the FED Document

This Feasibility Evidence Document includes NDI/NCS interoperability analysis results, business case (beginnings, including benefits analysis), risks assessment, process feasibility, architecture feasibility. This document is the simplest criteria to introduce feasibility. It shows that the feasible of the project and the project can be done on schedule without delay and within budget.

1.2 Status of the FED Document

This is the first version of FED document. Basically, this document includes business case analysis, architecture feasibility analysis, process feasibility analysis, risk assessment and NDI/NCS Interoperability analysis.

2. Business Case Analysis

Table 1: Business Case Analysis

Assumptions: • People using iPhone and have internet. • The project should have incentive for user to join in. • Users would like to show their posts. • Users would like to challenge their friends.			
Who	**What**	**Why**	**For Whom**
• Developers • Clients • Maintainers • Users	• Develop the challenge part of the app • Maintain the app • Keep using the app and give feedback • Challenge their friends	• Attracting more people to join the activities • To promote challenge activities • To help students to have fun in their daily life	• Users • Clients
Cost (Cost factors) • Maintenance costs • Development costs		**Benefits** (Key performance indicators - KPIs) • Increase the number of users in "Populic" • Increase the revenue in the future • Increase the posts	

2.1 Cost Analysis

The cost is the development cost (effort and time), maintenance cost and negotiation with client cost (time)

2.1.1 Personnel Costs

Table 1: Personnel Costs

Activities (12 weeks in total)	Time Spent (Hours)
Exploration, Valuation and Foundation Phases	
Client: win win negotiation with client session 1	2
Client: win win negotiation with client session 2	2
Client: meeting via email, slack	5
Architecture Reviews Boards	6
Development and Operation Phases	
Development (2hrs/week * 12 * 1person)	24
Project Process Meeting with Client (1hr /2weeks * 12 weeks)	6
Total	45
Maintenance Period (1 year)	
Maintenance	1/week
Total	52

2.1.2 Hardware and Software Costs

Table 2: Hardware and Software Costs

Type	Cost	Rationale
Mac	Free	Development operating system
React-Native	Free	Framework and API dependency
Xcode	Free	Development tools
WebStorm	Free	Development tools
AWS	$200/month	Server
Firebase	Free	DBMS

2.2 Benefit Analysis

The benefits of the project are these:

- Increase the number of users
- Increase the posts
- Increase the revenue in the future
- Save upload time

As a new and funny way to get more posts from users of “Populic”, the challenge game will make the “Populic” project more success by sending challenge tasks to friends.
As a nonprofit project, it is very hard to calculate the benefits of this app. Based on qualitative, benefits of “Populic” can be summarized as attracting more people to use the app through inner challenge game.
We considered that if we can decrease to the time of uploads, then, we will be able to attract more people to use our project and increase the posts after a certain time. Suppose daily post was

50 posts and in the first 5 years, assuming every day, we had 50 uploads and currently each post will cost 40s to upload to server.

Table 3: Benefits of Challenge System

Current activities & resources used	% Decrease	Number increased (Number/Year)
Uploads time saved		
Decrease the time of uploads(Currently it took about 40s to upload a picture and we assumed that we have 50 uploads per day, and it will totally take 5 hours for uploading per day, so 40*5*365 = 203 hours)	100%	101.5
Total		**101.5**

2.3 ROI Analysis

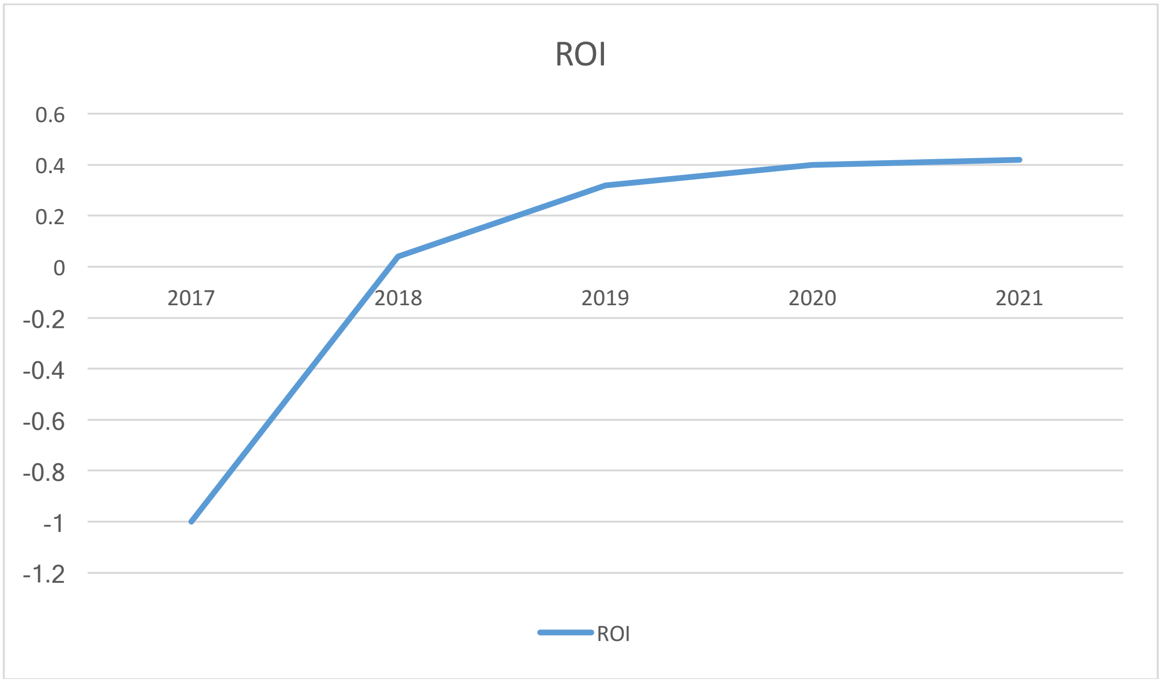
As there is no existing current business model, to calculate ROI is extremely hard. The client total time cost for 2017 is approximately 45 hours. For the following 4 years, we assume that the app maintenance will increase 10% per year.

There will be a fixed payment of $200 per month for Amazon AWS.

Table 4: ROI Analysis

Year	Cost	Benefit (Time Saved)	Cumulative Cost	Cumulative Benefit	ROI
2017	45	0	45	0	-1
2018	52	101.5	97	101.5	0.04
2019	57.2	101.5	154.2	203	0.32
2020	62.92	101.5	217.12	304.5	0.40
2021	69.212	101.5	286.332	406	0.42

Figure 1: ROI Analysis Graph



3. Architecture Feasibility

3.1 Level of Service Feasibility

Table 5: Level of Service Feasibility

Level of Service Requirement	Product Satisfaction
LOS-1: The layout of challenge game should be responsive and at least support all size of iPhone	Product Strategies: Xcode, React-Native API package
	Process Strategies: Set proper size based on the window size, and React-Native stylesheet will judge the size of layout.
	Analysis: We will use Xcode simulator to test the visibility of the challenge game to see if the challenge in different iPhone device has the same performance.
LOS-2: The challenge game should be able to load data less than 2 seconds	Product Strategies: Xcode, React-Native API package
	Process Strategies: Instead of fetching all data in one time, only fetch the data user needed. When user jump to a specific pop up page, the page will atomically download data user currently needed but not all data of all pages. So that the load time has been decreased.
	Analysis: We installed the app in our own phones, and test the loading time of all pop up pages in different Internet environment. With this new strategy, the load time for pop up page was less than 1 second.

3.2 Capability Feasibility

Table 6: Capability Requirements and Their Feasibility Evidence

Capability Requirement	Product Satisfaction
CR-1: Challenge photos and videos post	Software/Technology used: Xcode/WebStorm, React-Native libraries.
	Feasibility Evidence: Develop a prototype to implement challenge function based on React-Native Framework and API dependency
	Referred use case diagram:
CR-2: Challenge game pop screen	Software/Technology used: Xcode/WebStorm, React-Native libraries.
	Feasibility Evidence: Develop a prototype to implement a popup page which is a main page of the challenge game based on React-Native Framework and API dependency

	Referred use case diagram:
CR-3: Time complete	Software/Technology used: Xcode/WebStorm, React-Native libraries.
	Feasibility Evidence: Develop a prototype to display the remaining time of challenge and counting how long the user has spent on his/her challenge game based on React-Native API dependency
	Referred use case diagram:
CR-4: View, approve or cancel challenge	Software/Technology used: Xcode/WebStorm, React-Native libraries.
	Feasibility Evidence: Develop a prototype to implement function which user can check if his/her friend has finished their challenge based on React-Native API dependency.
	Referred use case diagram:
	Referred use case diagram:
CR-5: Challenge game suggesstion	Software/Technology used: Xcode/WebStorm, React-Native libraries.
	Feasibility Evidence: Develop a prototype to implement challenge game suggestion based on React-Native Framework
	Referred use case diagram:
CR-6: Further 5 days challenge content post	Software/Technology used: Xcode/WebStorm, React-Native libraries.
	Feasibility Evidence: Develop a prototype to implement further 5 days challenge content post based on React-Native Framework
	Referred use case diagram:
CR-7: Notification	Software/Technology used: Xcode/WebStorm, React-Native libraries.
	Feasibility Evidence: Develop a prototype to implement notification system based on React-Native Framework and API dependency
	Referred use case diagram:

3.3 Evolutionary Feasibility

Currently, no evolutionary requirement has been negotiated by the time of making this FED document version 1.1.

Table 7: Evolutionary Requirements and Their Feasibility Evidence

Evolutionary Requirement	Product Satisfaction
ER-1: << ER name >>	Software/Technology used:
	Feasibility Evidence
	Referred use case diagram:
	Software/Technology used:
	Feasibility Evidence:
	Referred use case diagram:
	Software/Technology used:
	Feasibility Evidence:
	Referred use case diagram:

4. Process Feasibility

Table 8: Rationales for Selecting Architected Agile Model

Criteria	Importance	Project Status	Rationales
30 % of NDI/NCS features	2	2	This app is mainly built using Xcode tools.
Single NDI/NCS	1	0	Single NDI/NCS may not provide us solution
Unique/ inflexible business process	1	1	Not unique
Need control over upgrade / maintenance	2	3	Need maintainer to update content of the challenge game
Rapid deployment	2	2	Client provide appropriate NDI solution for us to implement the project.
Critical on compatibility	1	2	The first version of the project is only design for iPhone(IOS 10+)
Internet connection independence	2	1	Need internet connection
Need high level of services / performance	4	1	As a social app, the project should allow multiple users online at the same time
Need high security	2	2	Username and password are required by app before user login app challenge game.
Asynchronous communication	2	2	Using firebase to handle asynchronous communication.
Be accessed from anywhere	2	2	The app needs web services
Critical on mass schedule constraints	3	3	Have to adhere one semester timeline
Lack of personnel capability	2	2	First of all, the user interface is simple and friendly, even the client who was a non-technical person can easily use it. Secondly, we have provided a detailed user manual to guide user to use our project.
Require little upfront costs	3	3	Currently, all development tools and libraries are open

			source, but AWS will require client $200 per month
Require low total cost of ownership	3	3	AWS will require client $200 per month and this will be the total cost currently.
Not-so-powerful local machines	3	3	The app doesn't need a powerful machines to run it

5. Risk Assessment

Table 9: Risk Assessment

Risks	Risk Exposure			Risk Mitigations
	Potential Magnitude	Probability Loss	Risk Exposure	
Inaccurate understanding of requirements: The client asked that user can challenge his/her app friends, however there is no friend system in the app and user can only join in community and follow other people and currently, even the client cannot give a specific explain about what is "friend". Also, there is no friend list in the app and it is impossible for user to challenge friends via friend list.	5	6	30	Verify win condition with clients, currently, we assume that if people followed each other, then, they will be considered as friends. Incremental development the highest priority features and functions first, negotiate with the client to see if we should add a friend system or use the above concept to define friend.
Requirement Changes: For the score system, the initial idea is to give a rank list to show people who has the highest score on it. However, the later requirement changed from the rank list to user can use the score to buy some fancy stickers from inner app. For now, client changes his mind again and there is no specific requirement for this score system and the client is still thinking and evaluating about this part. Then, the source code of this part has been changed again and again.	5	7	35	Follow the incremental development strategy, to develop the important part first to make sure that the main functions of the challenge game are working fine. Leaving the unsure parts away and give a suitable connector source code for it. To negotiate with the client and also give some possible choices for the unsure parts to client to think about.
Personnel shortfalls: Since React-Native is a new technology for us, many team members have no	3	8	24	Doing researching and self-learning for the new technology and to understand how to use the React-Native

experience on it. Besides these, this is a new team and we don't know each other very well, sometimes there is a communication problem between members of diverse background.				framework to build app. To schedule team meetings as more as possible and have more communications and negotiations with the members to better understand their backgrounds and skills.
Software interface mismatch: During the development, the client changes the idea of UIs, fonts. colors and layout design again and again and never has a final version of the layout.	6	7	42	Making several prototypes of the interface layout and negotiate with client to figure out which version will have satisfied with him. And also negotiate with client to suggest him to hire people who are good at UI design instead of designing UI by himself and change it again and again later. And also, based on our knowledge, give client some suggestions about the choice of background color and font.

6. NDI/NCS Interoperability Analysis

6.1 Introduction

We chose NDI products based on what we need during the development. For our project, we used Xcode and WebStorm as development tools to create the interface and layout of the challenge game, such as buttons, showing text content and Xcode modulator can be used to view if each component of the app is in the correct position. Also we use Firebase to manage data and this DBMS can directly store data as Json format. Finally, we use Amazon AWS as our server.

6.1.1 COTS / GOTS / ROTS / Open Source / NCS

Table 10: NDI Products Listing

NDI/NCS Products	Purposes
Firebase	DBMS
AWS	Server
React-Native	Framework
Xcode/WebStorm	Development tool

6.1.2 Connectors

In this project, we use Node.js/Firebase Connector to enable the web application to retrieve and query data from the database.

6.1.3 Legacy System

Currently, we do not have Legacy System.

6.2 Evaluation Summary

We will use the React-Native API which will offer us dependency framework packages and Firebase which can store Json format data as the DMBS and AWS as the server.

Table 11: NDI Evaluation

NDI	Usages	Comments
AWS	Server	• High performance • High security • Easy to configure
React-Native	Framework	• Support both iOS and Android development

		• Easy to learn and use if you familiar with JavaScript • Open source
Firebase	DBMS	• High performance • Can store Json format data • Open source
Node.js	Programming language	• Package ecosystem is the largest open source libraries in the world • Provides a non-blocking I/O API