Gifter Software Development Plan (SDP)

4.1 Software Development Plan Introduction

This Software Development Plan provides the details of the planned development for Gifter, an iOS application for tracking users’s wish lists. Sharing the cost of everyday things such as cab rides has becomes very simplistic through applications from Uber and Lyft. This functionality should be brought to everyday purchases of goods. Moreover, these items should be visible to people that are most likely to purchase items for a user such as their friends or family. Gifter will provide users a way to save favorited items, view friends profiles and their wish lists, and purchase items for friends with an option to split the cost with other friends.

Gifter Milestones		
Item	**Description**	**Due Date**
D#01	“Hello world” client request to API and Neo4j database setup	Week 8
D#02	Connect to Amazon API Add user and user credentials from client to database	Week 9
D#03	Search for items through Amazon API	Week 10
D#04	Add item to personal wish list	Week 11
D#05	Search for friends Request to add searched-for user to friends list	Week 12
D#06	View a friend’s wish list	Week 13
D#07	Purchase an item on a friend’s wish list for a friend	Week 14
D#08	Split cost of item on a friend’s wish list with another friend	Week 15
D#09	Code cleanup Debugging	Week 16 - Completion Date

4.1.1 Project Deliverables

The deliverables of Gifter include, but are not limited to, the following: an iOS Xcode project, a database model for the Neo4j database, and the source code of the RESTful web service.

4.1.1.1 iOS Xcode Project

The iOS Xcode project will contain all of the source code and dependencies required to install the Gifter application on an iOS device running iOS 8 and above.

4.1.1.2 Database Model

The database model will include all of the relationships possible in a Neo4j instance. Specifically, the model will contain the relationships between users-to-users and users-to-items.

4.1.1.3 RESTful Web Service Source Code

The web service source code will come in the form of a directory hierarchy that starts at the top level listener and works its way down to the scripts that communicate with the Neo4j database instance.

4.2 Project Resources

In order to get Gifter to a stable 1.0 version, it will require many resources. Most of these resources fall under the software category, but it will still require some hardware.

4.2.1 Hardware Resources

Gifter's has a wide variety of requirements for its hardware resources. Because it will target something as specific as iOS for the client-side application, then it will require Apple hardware. The server-side application can be run on a basic Ubuntu machine or any flavor of the Linux operating system.

4.2.1.1 Development Environment

The development environment for the client-side application requires both a Mac computer and at least one iOS device. Developing anything in the Apple ecosystem requires Xcode for development, and it is only available on an Apple computer. While it provides simulators for any of the iOS devices available, a physical iOS device is required to test independent variables such as performance under various phone service carriers and wifi hotspots.

The development environment for the server-side application will be done on an Ubuntu virtual machine. This is exactly how it will be running in production as well. For performance purposes, the virtual machine will require at a minimal level the following: quad-core Intel processor at 3ghz, 32 gigabytes of ram, and 4 enterprise-class terabyte hard drives for the database instance.

The production environment, in order to maintain consistency, will be a replication of the development environment. This will assist in eliminating potential bugs when rolling out a new release of the application.

4.2.2 Software Resources

4.2.2.1 Operating Systems

- Ubuntu v14.10: This is the operating system that will host the Node server-side application.
- Mac OS X Yosemite 10.10.2: Yosemite is the most recent and stable version of OS X that will run the Xcode IDE.

4.2.2.2 Text Editors / IDE

- Sublime Text 2: This will be primarily used in the development of the Node RESTful web service
- Xcode v6.1.1: This version of Xcode was built for iOS 8, Gifter's target iOS version.

4.2.2.3 Languages / Plugins

- Swift v1.1: Swift is the new language for developing iOS applications.
  - Alamofire v1.1: Alamofire is open-source RESTful request library that streamlines the request cycle.
- Node.js v0.12.0: Node is a Javascript runtime environment used for building fast, scalable network applications.
  - Express v4.12.2: Express is the web framework used for building Node RESTful applications.
  - Neo4j v2.0.0: Neo4j is the REST api client used to connect Node code to a Neo4j instance.

4.2.2.4 Data Storage

- Neo4j v2.2 Community Edition: The community edition of the graph database will store the user data. The enterprise edition is overkill for the initial version of Gifter.

4.3 Project Organization

The development team behind Gifter will be divided into two teams: the API team and the mobile client team.

4.3.1 API Sub-Team

The API Sub-Team’s duty would be to perform the necessary request to Gifter’s Node REST service. This team will implement the paths by which the client can access. They will also create, read, update and delete the data in the Neo4j instance through the methods in the API.

4.3.2 Mobile Client Sub-Team

The Mobile Sub-Team is responsible for creating a user interface on the iOS platform. The team will use the Swift programming language to pass and receive data to the API by making HTTP requests. They will also display the information in a structured way that adheres to Apple’s iOS interface guidelines.

4.4 Project Schedule

This section provides schedule information for the Gifter project. The full details have been broken down below.

4.4.1 PERT / GANTT Chart

Task	Start Date	Finish Date	Total Hours
Watch Swift Tutorial	3/10/2015	3/12/2015	5
Create Signup View	3/13/2015	3/13/2015	2
Connect Signup View to API	3/14/2015	3/14/2015	3
Test Connection to Amazon API	3/14/2015	3/14/2015	4
Create Search for Item View	3/15/2015	3/15/2015	2
Connect Search for Item View to API	3/15/2015	3/15/2015	5
Create Wish List View	3/15/2015	3/15/2015	1
Connect Wish List View to API	3/15/2015	3/15/2015	2
Create Friends Search View	3/18/2015	3/18/2015	2
Connect Friends Search View to API	3/18/2015	3/18/2015	3
Create Purchase View	3/29/1015	3/29/1015	3
Connect Purchase View to API	3/29/1015	3/29/1015	3
Create Item View	3/29/1015	3/29/1015	3
Connect Item View to API	3/29/1015	3/29/1015	3
Review and Test the Credit Card API	3/30/2015	3/30/2015	4
Create Request Split Cost View	3/31/2015	3/31/2015	4
Connect Create Request Split Cost View to API	3/31/2015	3/31/2015	6
Create User Settings View	4/1/2015	4/1/2015	3
Connect User Settings to API	4/1/2015	4/1/2015	3

4.4.2 Task / Resource Table

Task	Resource	Total Hours
Watch Swift Tutorial	Mobile Team, 2-3 computers	5
Create Signup View	Mobile Team, 2-3 Macs	2
Connect Signup View to API	Mobile Team, 2-3 Macs API Team, 2-3 Ubuntu machines	3
Test Connection to Amazon API	API Team, 2-3 Mac or Ubuntu machines	4
Create Search for Item View	Mobile Team, 2-3 Macs	2
Connect Search for Item View to API	Mobile Team, 2-3 Macs API Team, 2-3 Ubuntu machines	5
Create Wish List View	Mobile Team, 2-3 Macs	1
Connect Wish List View to API	Mobile Team, 2-3 Macs API Team, 2-3 Ubuntu machines	2
Create Friends Search View	Mobile Team, 2-3 Macs	2
Connect Friends Search View to API	Mobile Team, 2-3 Macs API Team, 2-3 Ubuntu machines	3
Create Purchase View	Mobile Team, 2-3 Macs	3
Connect Purchase View to API	Mobile Team, 2-3 Macs API Team, 2-3 Ubuntu machines	3
Create Item View	Mobile Team, 2-3 Macs	3
Connect Item View to API	Mobile Team, 2-3 Macs API Team, 2-3 Ubuntu machines	3
Review and Test the Credit Card API	API Team, 2-3 Ubuntu machines	4
Create Request Split Cost View	Mobile Team, 2-3 Macs	4
Connect Create Request Split Cost View to API	Mobile Team, 2-3 Macs API Team, 2-3 Ubuntu machines	6
Create User Settings View	Mobile Team, 2-3 Macs	3
Connect User Settings to API	Mobile Team, 2-3 Macs API Team, 2-3 Ubuntu machines	3

Table of Contents