

Project Proposal

**12 February 2015**

**Team3:**

Derek Grove

Douglas Harshberger

Dylan Small

Evan Connors

Michael Nazzario

Table of Contents

**I. Introduction 2**

*I.I → Purpose*

*I.II → Scope*

*I.III → Rationale*

*I.IV → Major Constraints*

**II. Project Organization 3**

*II.I → Process Model*

*II.II → Organizational Structure and Responsibilities*

*II.III → Base-lined Effort Schedule*

**III. Team-specific aspects 5**

*III.I → Management Objectives and Priorities*

*III.II → Team Availability*

*III.III → Team’s Range of Skills and Experience*

**IV. Overview 6**

*IV.I → Use Case Model*

*IV.II → Use Case Scenarios*

*IV.III → Constraints*

*IV.IV → Risks*

*IV.V → Assumptions and Dependencies*

**V. Requirements 9**

*V.I → Software Requirements Specification*

*V.II → Functional Requirements*

*V.III → Requirement Traceability Matrix*

*V.IV → Nonfunctional Requirements*

**VI. Glossary 13**

*→ Terms and Definitions*

I. Introduction

***.I Purpose:***

The purpose of the Japanese Flash card application is to provide a better way for our client’s students to study Japanese. By updating and repairing the app while also adding functionality we can ensure a more pleasant experience than what was previously offered. We intend to create an easy way for the client to change the content of the app as well as add more functionality for the end user.

***.II Scope:***

Japanese Flash Card app will be utilized by Dr. Yongtaek Kim and given to his students to be used as a tool to study the Japanese language in a flash card system. The app will be able to display an image and give options for the user to select the answer that best describes the image. There will be support for audio to help with the answering process. There will be functionality added to allow the client to manipulate the database in a more user-friendly manner. We will also be changing the language from Kanji to Hiragana at the request of the client. At this time there is no functionality for user logins however this may be implemented.

***.III Rationale:***

The purpose of this project is to test the end user on their knowledge of the Japanese language. We need for the end user and the client to be able to easily navigate through the app. We also must update this app to the client’s specifications by adding additional chapters and fixing the text element so that it displays the correct Japanese symbols for each image.

***.IV Major Constraints:***

If we stick to the Android app we have the issue of only being offered on Android systems. If we choose to create this into a web app, one constraint would be that we need to find and set up the hardware to host the app. We would also need to find out how many users will be accessing the app and set up a login system. There is a time constraint however in which we have fourteen weeks to complete the project.

II. Project Organization

***.I Process Model:***

* Requirements Iteration
  + Feb 12th - Project Plan & Software Requirements Specification
* Analysis Iteration
  + Feb 26th - Software Design Specification v1, High-Level Design
  + Feb 19th - Planning Checkpoint Review
* Design Iteration
  + Mar 26th - Software Design Specification v2, Detailed Design
  + Mar 5th - Prototype
* Implementation Iteration
  + Mar 23rd - Beta
  + Apr 4th - Verification and Validation Results
  + Apr 23rd - Product Release
* Post-Delivery Iteration
  + Apr 30th - Legacy documentation and Final Report

***.II Organizational Structure and Responsibilities:***

Derek is the Business System Analyst Team Leader, his responsibilities include:

* POC for any problems/concerns with the project
* Organization and publishing of team deliverables
* Planning/scheduling and reviewing/editing of documents; Operations

Doug is the Technical Platform Specialist, his responsibilities include:

* System logic and data flows
* Engineering of components/objects of the system
* Glass-Box Testing

Dylan is the Lead Architect, his responsibilities include:

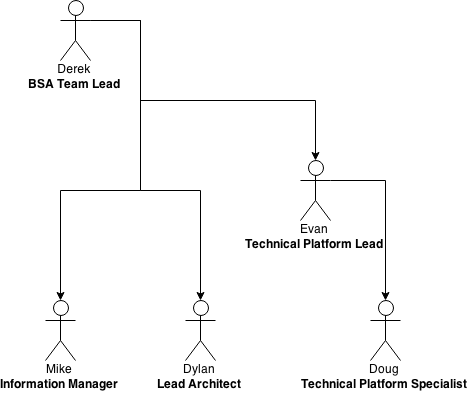
* Development of architecture documentation
* GUI and event module integration and management
* Black-Box Testing

Evan is the Technical Platform Team Leader, his responsibilities include:

* System logic and data flows
* Engineering of system structure and hierarchy
* System management, API documentation

Mike is the Information Manager, his responsibilities include:

* Data Entry and Documentation
* Manual/Minimal-programmatic changes
* Hardware/OS integration and validation



***.III Base-lined effort schedule:***

|  |  |
| --- | --- |
| **Date & Task** | **Hours** |
| Project Plan | 3 |
| Software Requirements Specifications | 2 |
| Software Design Specification V1 (High Level Design) | 5 |
| Planning Checkpoint Review | 1 |
| Software Design Specification V2 (Detailed Design) | 4 |
| Prototype | 10 |
| Verification and Validation Plan | 2 |
| Beta | 5 |
| Verification and Validation Results | 4 |
| Product Release | 2 |
| Legacy documentation and Final Report | 1 |
| **Total Hours** | 39 |

III. Team-specific aspects

***.I Management Objectives and Priorities:***

Our top initiative is to have a fully documented and working product at the end of the Spring 2015 semester. We will assess our progress at the end of each week with scorecards in which we will highlight achievements and outstanding items that may carry onto the following week. We plan to follow the Agile process and follow good programming practices, and abide by all SE codes of conduct.

***.II Team Availability***

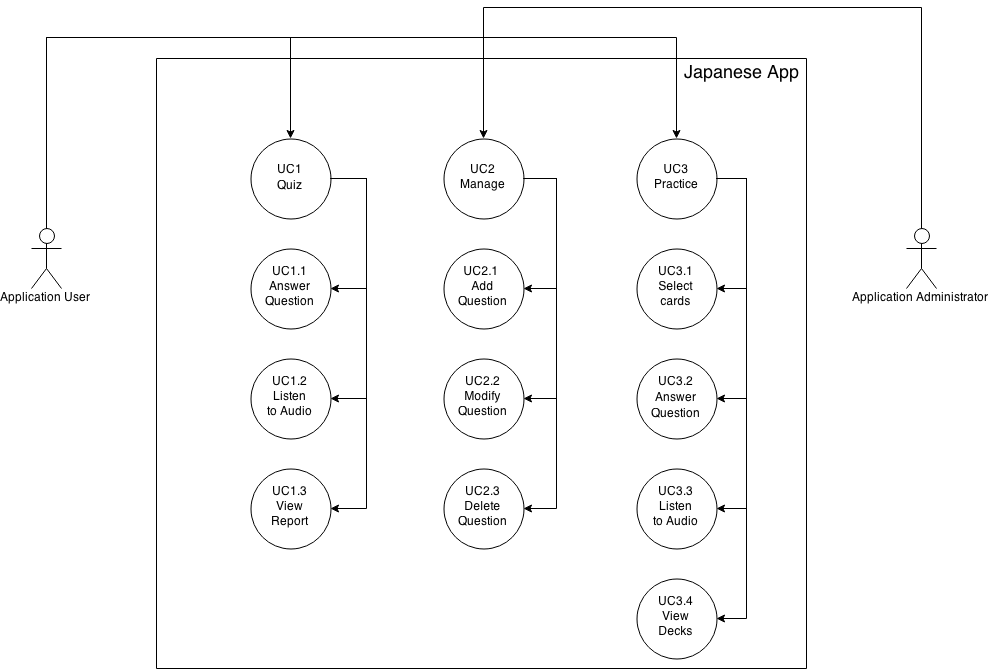
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Monday** | **Tuesday** | **Wednesday** | **Thursday** | **Friday** |
| **Derek** | After 6:00PM | After 7:00PM | After 6:00PM | After 7:00PM | After 3:00PM |
| **Douglas** | After 2:30 PM | After 6:30PM | After 12:30PM | After 6:30PM | After 12:30PM |
| **Dylan** | After 3:30PM | After 4:30PM | After 3:30PM | After 4:30PM | After 3:30PM |
| **Evan** | After 2:30PM | After 3:30PM | After 12:00PM | After 3:30PM | After 3:00PM |
| **Michael** | After 4:30PM | After 9:30AM | After 3:30PM | After 9:30AM | After 3:30PM |
| **Group Meeting** | 4:30 PM | 6:30 PM | 3:30 PM | 6:30 PM | 3:30 PM |

***.III Team’s Range of Skills and Experience***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Skill (1-5)** | **Derek** | **Doug** | **Dylan** | **Evan** | **Mike** |
| Java J2SE | 4 | 4 | 2 | 4 | 4 |
| Human-Computer Interaction (User interface) | 3 | 2 | 4 | 4 | 4 |
| General Database | 3 | 4 | 4 | 4 | 3 |
| SQL | 3 | 3 | 3 | 4 | 3 |
| Team work | 3 | 4 | 4 | 3 | 3 |
| Team management | 4 | 3 | 4 | 3 | 3 |
| Time management | 5 | 3 | 4 | 3 | 4 |
| Professional writing | 4 | 3 | 4 | 2 | 3 |
| Editing/Proofreading | 4 | 3 | 4 | 4 | 3 |
| Debugging/Troubleshooting | 3 | 2 | 2 | 4 | 3 |

IV. Overview

***.I Use Case Model:***



***.II Use Case Scenarios:***

|  |
| --- |
| **UC01: Quiz** |
| **Brief Description**  This use case allows the app user the ability to answer questions in a weighted system and see their final score of a section. |
| **Step-by-Step Description**   1. App User opens up the app and selects “Quiz”. 2. App User will then be given a question after they choose a lesson.    1. App User then selects an answer for the question.    2. App User may listen to audio for the question.    3. App User may choose to “Check Answer”.    4. App User chooses to go “Next” or “Previous”. 3. App User answers all of the questions in selected section. 4. App User receives a final report. |

|  |
| --- |
| **UC02: Manage** |
| **Brief Description**  This use case allows the application administrator the ability to import, modify, and delete questions and answers into the app using SQLiteBrowser. |
| **Step-by-Step Description**   1. App Administrator opens up the app and selects the “Management” tab. 2. App Admin will then be able to modify the database that the app accesses.    1. App Admin adds a question after filling in all required attributes in order to instantiate the object.    2. App Admin selects a question in which to edit a property contained within.    3. App Admin selects a question in which to delete and presses the “Delete” tab. |

|  |
| --- |
| **UC03: Practice** |
| **Brief Description**  This use case allows the app user the ability to practice questions within the app. |
| **Step-by-Step Description**   1. App User opens up the app and selects “Practice”. 2. App User will then be given a question after they choose a lesson.    1. App User will select cards from a list in which to practice.    2. App User then selects an answer for the question, answer will show after given.    3. App User may listen to audio for the question.    4. App User chooses to go “Next” or “Previous”. 3. App User answers all of the questions in the cards selected for the deck. 4. App User may now see the cards that were put in “Wrong” and “Correct” categories. |

***.III Constraints:***

Due to the nature of the project there are a few constraints associated with our project. One is that the end-user must be using a smartphone using the Android Operating System. Another constraint is that the questions must be manually entered into the database using SQLiteBrowser.  Also, the app must be reasonably lightweight and able to be run on a variety of devices due to the amount of different Android Operating Systems.

***.IV Risks:***

There are many risks associated with developing an app that can be used on many different phones. The main risk is the end users’ hardware that they are using.  Due to the wide variety of Android phones, there will be varying speeds of processors and accompanying components which each phone will have.  Therefore the app must be able to run on the most basic phones and run quickly as well.

***.V Assumptions and Dependencies:***

Our app will be able to install on any Android phone via an Android APK file.  The app will use SQLite on the back end to control all of the data going into the questions and answers.  The user will be able to import the multiple components required to create a question object into the app and the app will auto-generate questions and answers from the file.

V. Requirements

***.I Software Requirements Specification:***

To get a working and editable version of the system, please follow the directions as described below:

1. Download Eclipse and Android SDK
2. Download SVN plugin for Eclipse.
   1. Click [Help] tab, and click on Eclipse Marketplace...
   2. Search for “Subversive - SVN Team Provider” in the marketplace
   3. Install plugin
3. Change Perspective to SVN Repository Exploring
   1. In the top right hand corner of the Eclipse window is the Perspective toolbar. It will currently be set to Java
   2. Click the “Add Perspective” button, and then click SVN Repository Exploring

|  |
| --- |
| Note: If you can’t find the Perspective toolbar, you can also complete steps a. and  b. by clicking on the [Window] tab, then Open Perspective, then SVN  Repository Exploring. |

1. Adding New Repository Location
   1. In the lefthand side of Eclipse will be the SVN Repositories window
   2. Click the New Repository Location button. This will open a new window.
   3. In the URL textbox, enter svn://keeper.nsm.iup.edu
   4. Enter credentials and Check the Save Authentication checkbox if you’d like.
   5. Click Finish.
2. Checking Out a Project
   1. While in the SVN Repository Exploring perspective still, click on svn://keeper.nsm.iup.edu to view projects listed on the server
   2. Right click the JAPN102\_Flashcard folder, and click Check Out.
   3. Switch back to the Java Perspective to see the project. You can now work on the project.
3. Commit (Pushing changes you’ve made locally to the server build)
   1. In the Java perspective, right click the JAPN102\_Flashcard project, or a specific .java file you would like to push. Hover over Team, and then click Commit.
   2. You can add a comment to the update you are adding. Checkmark whatever items you want to update. Click OK.
4. Update Local Project
   1. In the Java perspective, right click the JAPN102\_Flashcard project, or a specific .java file you would like to update. hover over Team, and then click Update.
   2. This will update your local files to the latest revision that is stored on the server
5. Update Local Project to an Older Revision
   1. In the Java perspective, right click the JAPN102\_Flashcard project, or a specific .java file you would like to update. hover over Team, and then click Update to Revision.
   2. Either choose a date, or pick a specific revision to revert back to.
6. Run the Android application

|  |
| --- |
| Note: There maybe a dialog box that comes up with an error while trying to find the  Android SDK. Point it to ADT package you downloaded, sub-directory: ‘SDK’ |

1. Highlight project in project view pane
   1. Go to [Project] tab → Click ‘Clean...’ → Select Project → [Ok]
2. Run again, should get a built version without errors. Another dialog box will pop-up telling you that it needs a virtual device to run off of, unless you have a VM set-up.
   1. Refer to next steps if this is you.
3. [Window] tab → ‘Android SDK Manager’
   1. Make sure that the ‘Show:’ checkboxes ‘Updates/New’ & ‘Installed’ are checked true
   2. Go through and check API packages you wish to run your device off of and other packages needed to get the VM running.

|  |
| --- |
| Note: You can simply select all packages to ease your mind of installing the correct  packages, however it will make your SDK package quite large. |

* 1. → click [Install packages…]
  2. Close out of Eclipse, there will be open dependencies, SDK Manager is a separate entity so it’s okay to close out.

1. Open Eclipse back up click [Help] tab → ‘Check for Updates’

|  |
| --- |
| Note: If you want to download all packages, then follow steps 12-13 for there will be pre-requisite and dependency packages. This takes about 3 iterations. |

1. [Window] tab → ‘Android Virtual Device Manager’
   1. ‘Device Definitions’ → Select a device you want to use
   2. Click ‘Create AVD...’
      1. Eclipse should auto-generate fields to satisfy a complete VM.
   3. [Ok]
   4. ‘Android Virtual Devices’ → you should now see your device, highlight and click “Start” and eclipse should put everything together to where when you hit play.
   5. Your Virtual Android Device should pop-up in a separate window.
      1. Once it has loaded, you can press play in Eclipse to download the application to your device.

***.II Functional Requirements:***

FR1: The system shall quiz students on their Japanese vocabulary words, give them the correct answer, and also give them a total score after completing the entire lesson.

FR2: The system shall keep a database of corresponding images, audio files, and Japanese equivalent words (in Hiragana), and false (trick) answers. Information will be organized according to class lessons (1 - 23).

FR3:The system shall allow the instructor, Dr. Yongtaek Kim, to add/delete/change content on database easily, so he can update the application to fit with class material.

FR4: The system shall allow students to use a practice mode.

***.III Requirement Traceability Matrix:***

|  |  |  |  |
| --- | --- | --- | --- |
|  | **UC1** | **UC2** | **UC3** |
| **FR1** | X |  |  |
| **FR2** |  | X |  |
| **FR3** |  | X |  |
| **FR4** |  |  | X |

***.VI Nonfunctional Requirements:***

NR0: The system shall have minimal downtime, and once completed, the system should be available 95 % of the time, with 5% given for maintenance, database updates, and unforeseen failures.

NR1: The system shall support Android phones and tablets using Android Versions 4.0.3(Ice Cream Sandwich) to 5.0 (Lollipop) when released.

NR2: The system shall be light-weight, and be able to run on cheaper Android devices with low processing ability.

VI. Glossary

**APK –** Android Application Package; Application packaging system used to distribute and install Android applications.

**CSV –** Comma Separated Values; Commonly used amongst professionals of all departments providing a clean and easy tabulated format.

**Hiragana -** the more cursive and more widely used form of kana (syllabic writing) used in Japanese, especially used for function words and inflections.

**Kanji -** a system of Japanese writing using Chinese characters

**Login system -** an element in the app that allows users to login with an ID and password to regulate who can access the app.