Turtl
Use Case Specification

Submitted to:

Asst. Prof. Ma. Rowena C. Solamo
Faculty Member
Department of Computer Science
College of Engineering
University of the Philippines, Diliman

Submitted by:
Nikki Balugay
Ram Mangaoang
Brian Sy

In partial fulfillment of academic requirements
for the course
CS 191 Software Engineering I
of the
1st Semester, AY <2017-2018>

Unique Reference:

The documents are stored in the https://github.com/brianpesy/turtlios.

https://github.com/brianpesy/turtlios/tree/master/02%20-%20Requirements%20Engineering

Document Purpose:

This document is for further explaining different functionalities within the program to a more specific and clear extent while also maintaining readability as well.

Target Audience:

Businesses, clients, and those people who are interested in the whole project and how exactly we are going to go about designing the whole project are within the scope of our target audience.

Revision Control

History Revision:

Revision Date	***Person Responsible***	***Version Number***	***Modification***
10/07/2017	Brian Nicholas Sy	1.0	Initial document, document purpose, target audience, use case name, description, preconditions, and diagram

Use-Case Name: 7.0 Note will be saved on the server - 7.0 Save Note

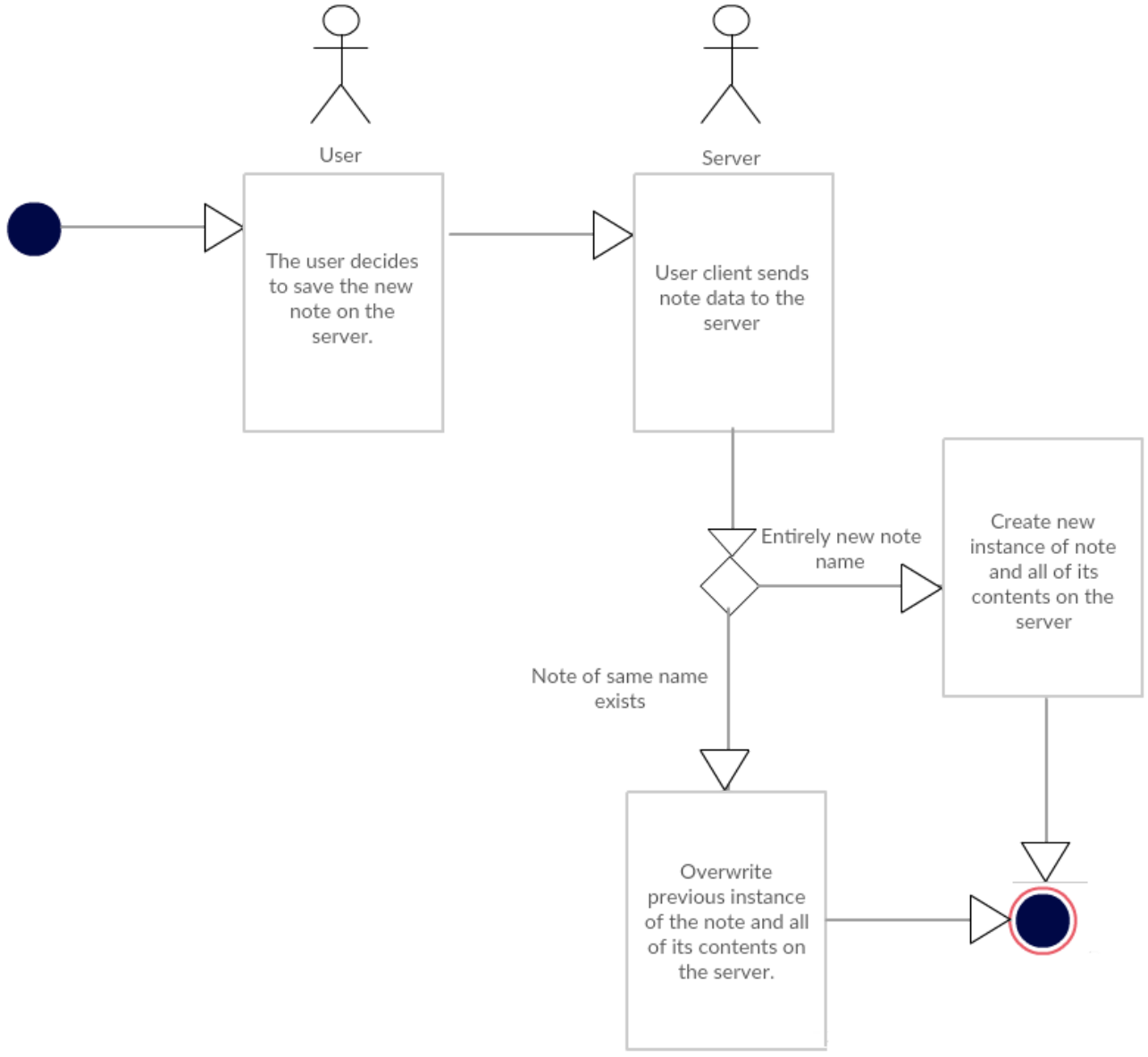
Description: For this use case, it will detail the process of saving notes. Here, we work with being able to save notes that were created by the user and then put them into the server for accessibility.

Preconditions: The user would need to have at least added a note or a pre-existing note to overwrite.

Flow of Events:

Scenario Name	***Description***
Scenario 1 (Basic Flow) The user saves an entirely new note.	1. The user decides to save the new note into the server. 2. The user's client will send the note data over to the server, and it will save the note. 3. Depending on whether or not there is a note of the same name (in this case, it is entirely new), it will create a new instance of the note and all of its contents on the server.
Scenario 2 The user saves an edited version of an existing note.	1. The user decides to save the new note into the server. 2. The user's client will send the note data over to the server, and it will save the note. 3. Depending on whether or not there is a note of the same name (in this case, there already exists one), it will overwrite the previous instance of the note and all of its contents on the server.

Activity Diagram of the Flow of Events:



Postcondition: The server will now hold the note, and it is saved.

Relationships: NONE

Special Requirements:

NONE