**Requirements Analysis Document**

Study Bear

CSCI 4712 Senior Capstone Project

Spring 2015

Georgia Regents University

Augusta, GA

Date: 2/1/15

Version 0

Team Members

Jeremy Scott

Chad Reynolds

Darcie Odom

**Abstract**

This document contains the requirements, analysis and design artifacts for Study Bear. Study Bear is a software system designed to aid its users in finding study partners for their class work.

The users will register for the system using an Android application. Once they have access, they will be able to enter or update information for the system to use to match them up with study partners. Matched users will be able to communicate using a message inbox system. Users will be also be able to manually search for other users.

This document describes the requirements, analysis and design of the Study Bear. The rest of this document is structured as follows. Chapter 1 contains the introduction. This chapter presents a brief description of the system. Chapter 2 outlines the functional requirements of the system.

**Table of Contents**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | INTRODUCTION…………………………………………………...... | | | 5 |
|  | 1.1 | OVERVIEW OF SYSTEM…………………………………………... | 5 |  |
|  | 1.2 | SCOPE OF SYSTEM………………………………………………… | 5 |  |
|  | 1.3 | OVERVIEW OF DOCUMENT……………………………………… | 5 |  |
| 2 | REQUIREMENTS OF SYSTEM…………………………………...…. | | | 6 |
|  | 2.1 | FUNCTIONAL REQUIREMENTS………………………………..….. | 6 |  |
|  | 2.2 | USE CASES…………………………………………………………….. | 7 |  |
|  | 2.3 | USE CASE DESCRIPTIONS………………………………………….. | 8 |  |
|  | 2.4 | REQUIREMENTS ANALYSIS………………………………………… | 14 |  |
|  |  | | |  |
|  |  |  |  |  |

# Introduction

## Overview of System

Study Bear is a system for finding study partners. Its actors are *Users*, the actual users of the system. The matchmaking system gives the functionality to find and plan with study partners with minimal effort on the part of the user.

Study Bear allows the users to register new accounts in the system, input and manage information about what classes they are taking and have taken, be matched automatically by the system with other users based on their information, and also search for other users manually by name and email. Study Bear is accessed through an Android capable device running the Study Bear application, and is supported by a backend server accessed via network connection.

## Scope of system

The matchmaking system provides the functionality for users to manage accounts and information to aid the system in finding study matches. The data store is externally located on a server accessed through a network. The system requires this server for data storage, processing and generating responses to user matchmaking and search requests, and out-of-band communication with the user for features like account validation and password resetting.

## overview of document

The rest of this document is structured as follows. Chapter 2 outlines the functional requirements of the system. Within this chapter is a list of functional requirements of Study Bear. It also includes a use case model of those functional requirements. A detailed description of each functional requirement then follows.

# requirements of system

## Functional Requirements

## Login -

## Logout -

## RegisterAccount -

## PasswordReset -

## ValidateUserAccount -

## SearchUsers -

|  |  |
| --- | --- |
| *Use case name* | FindStudyPartner |
| *Participating*  *actors* | User |
| *Flow of events* | 1. The User initializes use case by clicking on the “match” button in some screen.  2. StudyBear replaces the current screen with the FindStudyPartner screen that initially has a loading matches message and a back button. Then StudyBear sends a request to the server to find matches for the User.  The server ^searches the data store for matches to the User, returning the results of the search to StudyBear.  StudyBear displays the first match to the user.  3. The User has three different gestures to respond to the match: positive, negative, or block.  4. StudyBear temporarily stores the User’s response and reacts in one of three ways:  a. \*\*Positive - The SendMessage use case is initializes with the User sending a message to the match  b. Negative/Block - StudyBear displays the next match  c. Out of Matches - StudyBear sends the temporarily stored responses of the user to the server, [server stores] and then sends another request to find matches for the User.  5. The User clicks the back button to indicate they are done on the FindStudyPartner screen.  6. StudyBear sends the temporarily stored responses of the User to the server, and then closes the FindStudyPartner screen and displays some screen.  The server stores the User responses in the data store. |
| *Entry condition* | * The User has logged in to StudyBear (or currently viewing some screen). |
| *Exit condition* | * The User has started the SendMessage use case or is on some screen. |
| *Quality*  *requirements* | * The server should not spend more than x amount of time searching for matches. |

\*We should go through the mockups/use cases, determine all of the pages needed, and give them names for reference in our use cases.

^greatly simplified, but we did this last semester for the schedule generation

\*\*What should happen to set up a study date? Do we want to just go to sending a message?

|  |  |
| --- | --- |
| *Use case name* | PasswordReset |
| *Participating*  *actors* | User |
| *Flow of events* | 1. The User initializes use case by clicking the password reset button on the Login screen [enter email].  2. StudyBear sends the password reset request to the server and displays a message to the User that a password reset email is being sent. The server generates a reset code linked to the User’s account. The server sends an email with a hyperlink using the reset code for resetting the User’s password.  3. The User receives the email and visits the password resetting page.  4. The server displays a page based on the reset code in the hyperlink: [Should we have some sort of recovery question?],  an input box for entering a new password, and a button for submitting.  5. The User enters a new password [and maybe answers a recovery question] and clicks the submit button.  6. The server queries the database with the User submitted information and the reset code, verifying if the password should be reset.  a. If the server verifies the information, then the new User password is saved in the data store. The server displays a success message to the User on a new page.  b. If the server does not verify the information, then the server displays an error message to the User on a new page. |
| *Entry condition* | * The User is on the Login screen. |
| *Exit condition* | * The User’s password has been reset. |
| *Quality*  *requirements* | * Reset code should only be valid for a limited amount of time. |

\*Recovery question? Do those help?

\*\*Should we be using a webpage for the password reset? Would it benefit us to do it in the application? This decision is somewhat related to how we verify emails, if we make them visit a page or send an email back to verify.