# Revision History

## Revision History Details

<table>
<thead>
<tr>
<th>Rev</th>
<th>Date</th>
<th>Author</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10/20/2008</td>
<td>Charles Brexel</td>
<td>Initial Draft</td>
</tr>
</tbody>
</table>
# Table of Contents

1. Introduction ................................................................................................................ 1  
   1.1 User Management System  
   1.2 Database System  
   1.3 User Interface  
   1.4 File Transfer  
   1.5 Application Framework  
   1.6 Wiki Pages  

2. Implementation ............................................................................................................ 2  
   2.1 User Management System  
      2.1.1 Classes  
      2.1.1.1 user.php  
      2.1.1.2 admin.php  
      2.1.1.3 register.php  
      2.1.1.4 message.php  
   2.2 Database System ...........................................................................................  4  
      2.2.1 Tables  
      2.2.1.1 config  
      2.2.1.2 users  
      2.2.1.3 user_field  
      2.2.1.4 user_info  
      2.2.1.5 messages  
      2.2.1.6 applications  
   2.3 User Interface ...............................................................................................  6  
      2.3.1 Classes  
      2.3.1.1 index.php  
      2.3.1.2 profile.php  
      2.3.1.3 department.php  
      2.3.1.4 inbox.php  
      2.3.1.5 files.php  
   2.4 File Transfer .................................................................................................  7  
      2.4.1 Classes  
      2.4.1.1 getfiles.php  
   2.5 Application Framework ................................................................................  8  
      2.5.1 Classes  
      2.5.1.1 api.php  
      2.5.1.2 app_api.php  
   2.6 Wiki Pages  

3. Testing  ......................................................................................................................  9  
   3.1 Definition and Objectives  
      3.1.1 Unit Testing  
      3.1.2 Module Testing  

3.1.3 Integration Testing
3.1.4 Regression Testing ................................................................. 10
3.1.5 System Testing
3.1.6 Beta Testing

3.2 Schedule ......................................................................................... 11

3.2.1 Test Plan Schedule
3.2.2 Testing Schedule
3.2.3 Dependencies

3.3 Responsibilities............................................................................. 12

3.4 Equipment

1. Introduction

1.1 User Management System
   The User Management System will retrieve and update information about the users of Business Circle.

1.2 Database System
   The Database System will store all information about Business Circle. It will hold information about the servers being used, installation information, the user information, the content, and other information needed by individual applications.

1.3 User Interface
   The User Interface is how individual employees will interact with Business Circle. Each employee will have a profile page in addition to shared company information.

1.4 File Transfer
   File Transfer will allow users to view files on the company's shared network and click a link that will open that file on the client machine.

1.5 Application Framework
   The Application Framework is a set of public facing functions or tags that allow developers to create programs that will seamless fit into the Business Circle application.

1.6 Wiki Pages
   Each department will have its own section of wiki pages where company knowledge can be shared, updated, and used.
2. Implementation

2.1 User Management System

2.1.1 Classes

2.1.1.1 user.php

The User class is responsible for the majority of the management of the user information. The User class provides the functions to retrieve as well as update information about users from the database.

- get_uid()
- get_firstName()
- get_lastName()
- test_password(password)
- get_position()
- get_permission(action)
- get_field(field)
- set_firstName(first)
- set_lastName(last)
- set_password(password)
- set_position(position)
- set_field(field)

2.1.1.2 admin.php

The Admin class is also responsible for managing user information; however, it is also responsible for all the other actions that an administrator can do within the Business Circle application. The Admin class can interface with the database to change system wide settings and can manage multiple users.

- get_users()
- get_firstName(uid)
- get_lastName(uid)
- test_password(password)
- get_position(uid)
- get_permission(uid, action)
- get_field(uid, field)
- set_firstName(uid, first)
- set_lastName(uid, last)
- set_position(uid, position)
- set_field(uid, field)
- get_admin_permission()
- get_company()
- set_company(company)
- get_database_ip()
- set_database_ip()
- get_network()
- set_network()
2.1.1.3 register.php
The Register class is responsible for adding new users to the system. It will create a new user in the table and utilize the functions of the user.php class to include new information.
  ● set_permissions(uid, permissions)
  ● create_user(user_info)
  ● validate_info()

2.1.1.4 message.php
Allows users of Business Circle to send private messages to each other over the intranet system.
  ● get_messages(uid)
  ● send_message(from_uid, to_uid, message)
  ● get_message(mid)
2.2 Database System

2.2.1 Tables

2.2.1.1 config
The config table will hold all the settings about the Business Circle application that are system wide.
- Company Name
- Application Server
- Database Server
- Description
- Mailserver
- Mail username
- Mail password

2.2.1.2 users
The user table will hold all the basic information about the users
- uid
- first_name
- last_name
- position
- permissions
- email

2.2.1.3 user_fields
The user_fields table will hold all the extra fields that an administrator can define.
- ufid
- name
- type
- required

2.2.1.4 user_info
The user_info fields table will hold all the information for the extra fields that are defined in the user_fields table
- uinfoid
- ufid
- uid
- info

2.2.1.5 messages
The messages table holds all the messages sent back and forth between users.
- Mid
- from_uid
- to_uid
- message
- timestamp
2.2.1.6 applications

The applications table holds all the information about additional applications that have been installed into the Business Circle application.

- Name
- author
- author_url
- file_directory

2.2.2 Classes

2.2.2.1 database.php

The Database class will abstract out all the queries to the database. Other classes will call functions in the database class instead of directly communicating with the database. This will allow the database structure, format, and protocol to change with minimal code changes.

- get_users()
- get_firstName(uid)
- get_lastName(uid)
- test_password(password)
- get_position(uid)
- get_permission(uid, action)
- get_field(uid, field)
- set_firstName(uid, first)
- set_lastName(uid, last)
- set_position(uid, position)
- set_field(uid, field)
- get_admin_permission()
- get_company()
- set_company(company)
- get_database_ip()
- set_database_ip()
- get_network()
- set_network()
- set_permissions(uid, permissions)
2.3 User Interface

2.3.1 Classes

2.3.1.1 index.php
index.php is the main engine that will display the main pages. It must communicate with the database in order retrieve necessary configuration options and then place the required information into the necessary PHP/SMARTY tags. Store the information in a template and output the necessary HTML.

2.3.1.2 profile.php
profile.php is responsible for the display of individual user's profiles. It must determine which uid is being called, retrieve the information from the user.php class and display it using the predefined SMARTY template.

2.3.1.3 department.php
department.php is responsible for displaying each individual departments wiki pages. It must retrieve the department's wiki information, insert that into the SMARTY template engine and output the necessary HTML.

2.3.1.4 inbox.php
inbox.php is used to display private messages among members. The class must communicate with the database in order to retrieve the messages only pertinent to the current user and then insert them into the template and output the HTML.

2.3.1.5 files.php
files.php will display the files on the shared network. It must call the getfiles.php in order to retrieve all the files in the current working directory and then display links to these files in the template.
2.4 File Transfer

2.4.1 Classes

2.4.1.1 getfiles.php

getfiles.php will communicate with the shared network and retrieve files in the current directory.

- get_network()
- get_files(directory)
- get_files()
2.5 Application Framework

2.4.1 Classes

2.4.1.1 api.php
The api.php class is a base class that defines certain functions that all api's must use.

- get_database()
- set_variable(set_name, value)
- get_variable(set_name)

2.4.1.2 app_api.php
The app_api.php will extend the api.php class and add onto it for all the necessary functions to build an application on the Business Circle platform.

2.6 Wiki Pages
An open source wiki will be integrated into the system.
3. Testing

3.1 Definitions and Objectives

3.1.1 Unit Testing

Unit Testing is the testing of the smallest units of the system. Each individual function within a class can be unit tested. The Unit Testing should be done in a White Box manner, so that edge cases can be tested.

Unit Testing is accomplished through the help of a framework called SimpleTest (http://www.simpletest.org). SimpleTest allows for easy unit testing of a PHP application. The unit tests are written as an extension of a class called UnitTestCase. Actions are performed and asserted against an expected output.

The objective of Unit Testing is to make sure that the building blocks of the system each perform their tasks. If errors are caught in the unit testing phase, they are much easier and less costly to fix when compared to if these errors are caught later.

3.1.2 Module Testing

Module testing will employ a black box testing strategy and test the output of a given module given an input. In order to complete module testing before the integration stage, dependencies will have to be broken by simulating input from another module.

The objective of module testing is to ensure that the individual pieces of the system achieve their individual goal before attempting to bring the whole system together.

3.1.3 Integration Testing

Integration Testing will test the dependencies of the system. A mix of black box and white box testing will need to be performed. This testing is very similar to module testing; however, the dependent modules have now been linked so that no input is simulated.

![Diagram](image-url)
The objective of integration testing is to make sure that the modules can correctly link together to form a coherent system and that the communication between modules is occurring correctly.

3.1.4 Regression Testing

Regression testing will have to occur throughout the testing process in order to establish whether new additions to the system have caused problems elsewhere. Regressions of module tests and systems test need to be performed periodically as bug fixes and new modules are introduced to the system.

The objective of regression testing is to make sure that dependencies do not cause problems throughout the system. Regression testing also allows for errors to be caught early and therefore decreases the cost of error fixing.

3.1.5 System Testing

Systems testing will establish whether or not the application fulfills the requirements set forth in the requirements document. Test cases will be established against the requirements document and testers will ensure that the entire application is running as expected.

3.1.6 Beta Testing

Outside testers/potential users of the Business Circle application will try the system. This is good for usability, performance, and also error testing. A tester pool of 25 people will attempt to interact with the system and provide feedback.
3.2 Schedule

3.2.1 Test Plan Schedule

Each test plan will need to have test cases developed. The following dates are when these documents need to be completed by.

<table>
<thead>
<tr>
<th>Test Plan</th>
<th>Schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit Test Cases</td>
<td>Developed as functions are developed: Finished by 10/28/08</td>
</tr>
<tr>
<td>Module Test Cases</td>
<td>10/24/08</td>
</tr>
<tr>
<td>Integration Test Cases</td>
<td>11/10/08</td>
</tr>
<tr>
<td>Regression Test Cases</td>
<td>11/10/08</td>
</tr>
<tr>
<td>System Test Cases</td>
<td>11/20/08</td>
</tr>
<tr>
<td>Beta Test Cases</td>
<td>11/24/08</td>
</tr>
</tbody>
</table>

3.2.2 Testing Schedule

The following schedule is for when each phase of the testing should be completed by.

<table>
<thead>
<tr>
<th>Test Phase</th>
<th>Schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit Testing</td>
<td>Completed throughout development. Finished by 10/28/08</td>
</tr>
<tr>
<td>Module Testing</td>
<td>10/30/08</td>
</tr>
<tr>
<td>Integration Testing</td>
<td>11/12/08</td>
</tr>
<tr>
<td>Regression Testing</td>
<td>Completed throughout development/testing</td>
</tr>
<tr>
<td>System Testing</td>
<td>11/22/08</td>
</tr>
<tr>
<td>Beta Testing</td>
<td>11/26/08</td>
</tr>
</tbody>
</table>

3.2.3 Dependencies

All the test cases can not be carried out concurrently. Unit Testing within the module must be completed before module testing, module testing must be completed before integration testing, and integration testing must be complete before system testing and beta testing can begin.

There are also dependencies within the modules as seen in Figure 1.3.
3.3 Responsibilities

Since this is an individual project, all responsibilities with testing will be completed by Charles Brexel except for Beta Testing. For Beta Testing, Charles will recruit 25 potential users of the application and receive their comments about the system. For every other phase of testing, Charles Brexel will have to develop the test case documents and carry out the testing.

3.4 Equipment

There are not a lot of necessary equipment needed for the testing of Business Circle. Most testing can be completed on any computer or laptop with a Browser. The only additional equipment/technology needed is the installation of the SimpleTest framework.