1. **Introduction**

The purpose of this software design document is to provide a low-level description of the marketing guide application, known as The Modern Marketer. This document provides insight into the structure, as well as a picture of what has gone into the design of each component.it displays the class daigram,state diagram and the sequence diagram.This document aims to equip the reader with a solid understanding of the inner workings of the Modern Marketer app.

**1.1 Goals and Objectives**

The goal of The Modern Marketer is to equip the users with the tools they need to develop their brands, professional or personal, and develop their businesses in a digital world. In a world which is going Mobile-only, having an online presence for businesses is of the utmost importance.This is where the app comes in handy, as it seeks to aid businesses in building powerful branding and storytelling skills.

The final product must be easy on the eyes, intuitive, efficient and provide users with what they want with ease. All the information must be accessible in a step-by-step manner, so that the user can easily follow along without feeling overwhelmed. The user interface must be clear and have all the details that the user might need at that particular stage. Beyond these general principles, the application must also provide with the following functionalities:

* Support for offline users.
* Miscellaneous tips for improving your brand value.
* Search option.
* A bookmark feature.
* A feature for taking suggestions and complaints.
* A feature for asking queries

**1.2 Project Overview and Scope**

The Modern Marketer app consists of two primary components: a client-side application that allows the user to track and progress with the tools and information, in their own preferred way, and a server-side application which updates the data with the ever-changing technology, and keeps a track of user complaints and suggestions.

**Core Platforms:** The platforms which dominate the internet currently, and where the user has to imperatively work on are covered here, and form the base of the app.

**Miscellaneous Tips:** The various other channels which support content creation and can be used to build your network online, as well as various other marketing tips are covered here.

**CORE FEATURES**

1. **WELCOME :** Appears on opening the app.
2. **Core Social Media Platforms:** Covers a step by step guide to increase the brand, and further tools to increase your audience.
3. **Other Miscellaneous Tips:** Covers other ways to build your audience and develop the brand which are not covered in the Core Platforms, and are a great way for businesses to gain some leverage.
4. **Suggestions and Queries:** This section allows the user to provide any suggestions they may have related to the app, and any other marketing techniques they may have in mind, as well as any queries that they may have related to the information provided.

The features below are not guaranteed to be present in the final release of The Modern Marketer, but will be added as time permits. Due to their tentative nature, they will not be covered in this document.

**ADDITIONAL FEATURES**

1. Help Menu

* Displays a list of topic covering the different components of The Modern Marketer.
* Offers detailed information on each menu, etc.
* Can be accessed at any time.

2. Push Notifications

* A notification is sent when the app is updated with any new feature.
* Whenever, new content is added to the core channels, as a guide, a notification is sent.
* If a user has some incomplete tasks left in the app, and he has set a bookmark for the same, a notification will be sent as a reminder for the user.

3. Tutorial

* Offers a step-by-step guide through each feature and menu.
* Enables any user to quickly and easily take advantage of all of the features of the app.

**1.3 Definition, Acronym and Abbreviations**

Cohesion - Cohesion refers to the degree to which the elements of a module belong together. Thus, cohesion measures the strength of relationship between pieces of functionality within a given module.

Coupling - Coupling is the degree of interdependence between software modules, a measure of how closely connected two routines or modules are; the strength of the relationships between modules.

Session -A login session is the period of activity between a user logging in and logging out of a (multi-user) system.

Required Field - A field in the data set of a document which is required for successful document generation.

Sequence Diagram - A Sequence diagram is an interaction diagram that shows how objects operate with one another and in what order. It shows object interactions arranged in time sequence.

Class Diagram - A class diagram is a static diagram and represents the static view of an application. It is an illustration of the relationships and source code dependencies among classes in the Unified Modeling Language (UML).

State Diagram - State diagram is used to describe the behaviour of a single object in response to series of events in the system.

**1.3 SOFTWARE CONTEXT**

The Android app store will be having The Modern Marketer, free of charge. Funding should not be an issue, as there are no development and maintenance costs.

Any features that do not make it into the initial release will be part of the future development plans. Any features that might need to be added based on the user demands are also something to take into consideration.

**1.4 REFERENCES**

N/A

## **1.5 Overview of Document**

The software design document is divided into various sections with sub-sections:

1. Introduction - It highlights the purpose and scope of the document, also explains the acronyms that are used throughout the document.

2. System Architecture Description - It describes the major architectural decisions that are taken regarding the project , it explains various modules and functions that make the webpage. It also includes the Sequence Diagrams, Class Diagrams and State Diagrams along with User-Interface details.

3. Description of Components - Every component in the project is explained here in detail.

4. Reuse and Relationships to other Products - It lists the modules and functions that will be shared by multiple processes.

5. Design Decisions - It includes various designs being used in the project.

6. Pseudocode for Components

7. Appendix

**1.5 INTENDED AUDIENCE AND READING SUGGESTIONS**

While the software requirement specification (SRS) document is written for a more general audience, this document is intended for individuals directly involved in the development of The Modern Marketer.

Below is a brief overview of each part of the document.

1. Introduction

* A brief introduction of what The Modern Marketer app is all about, its goals and objectives, project scope, general details, etc.

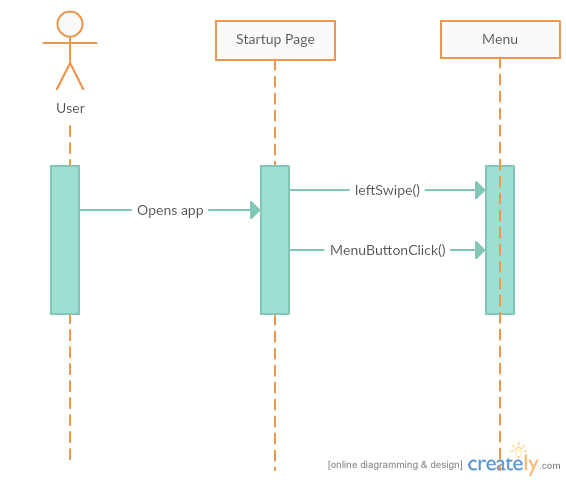
## **2.0 System architecture description**

### **2.1 Overview of modules / components**

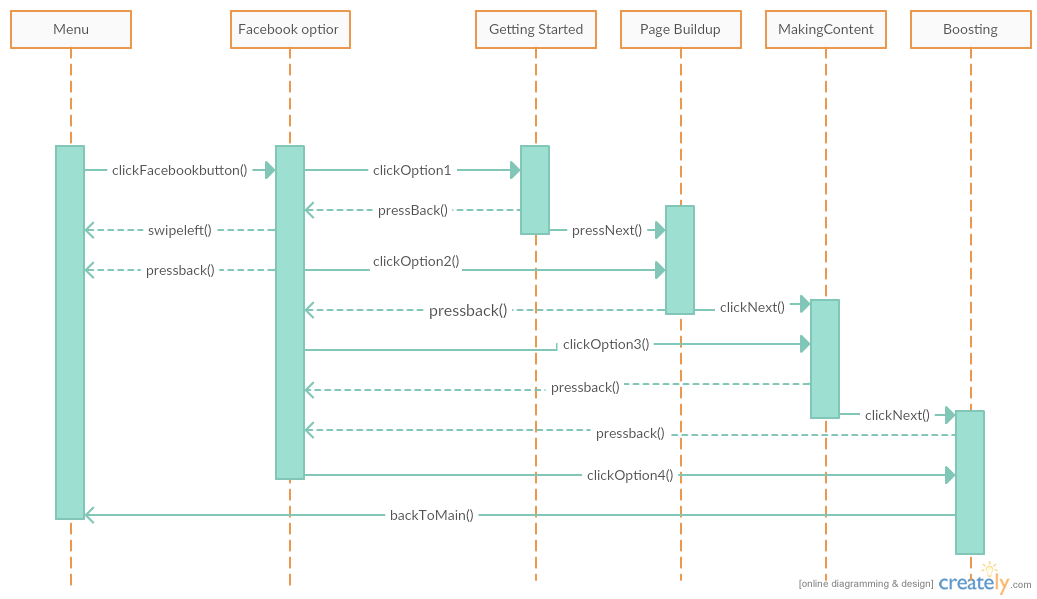
Our system is designed with extensibility and scalability in mind. We are taking great care in designing a framework which can be updated easily. Many of the anticipated changes to our system in future phases will only require adding new types of data and changing the user presentation code to make use of these new data. The framework we have designed will only require "plugging in" these new types of data without refactoring the logic that passes the data over the network, retrieves and updates the database, etc. There are five basic, logical components of the system: the Database Engine, the Class Library, the Server Procedure Proxy, the Server Application, and the Client Applications.

**2.2 Sequence Diagram**

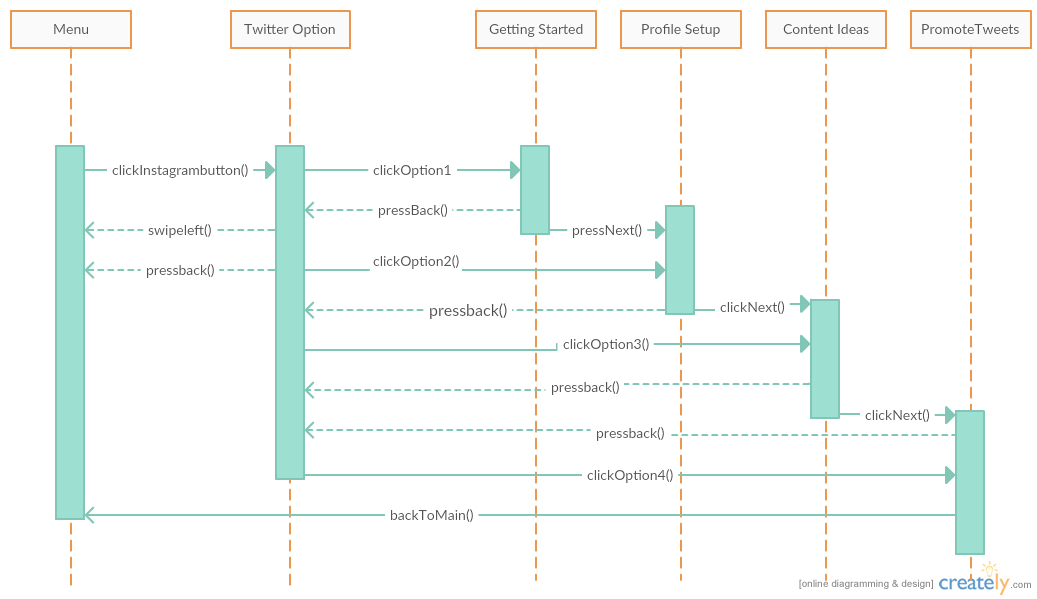
**1.menu1:-**



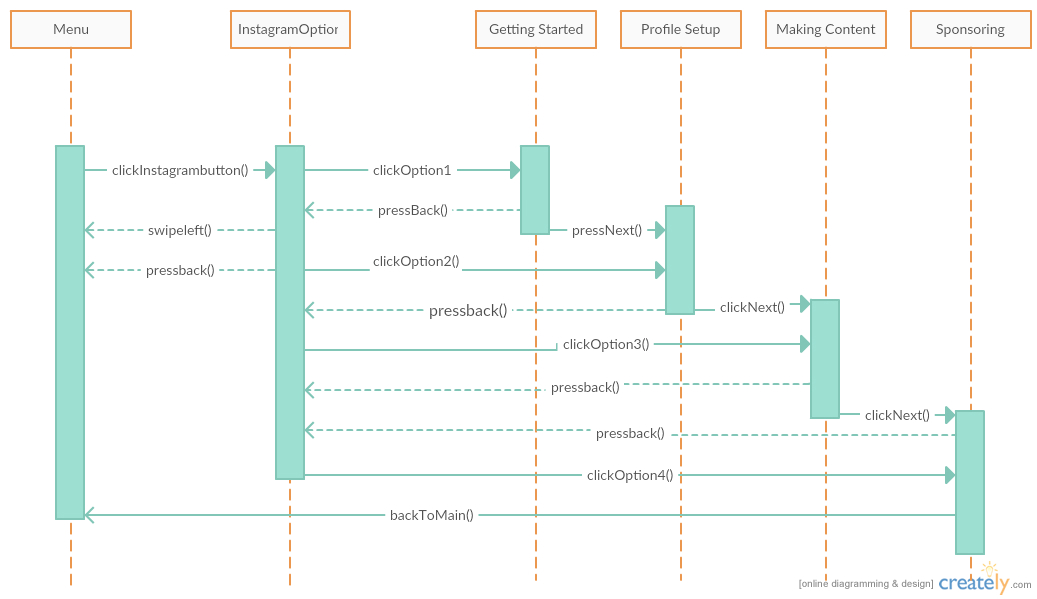
**2.facebook menu**



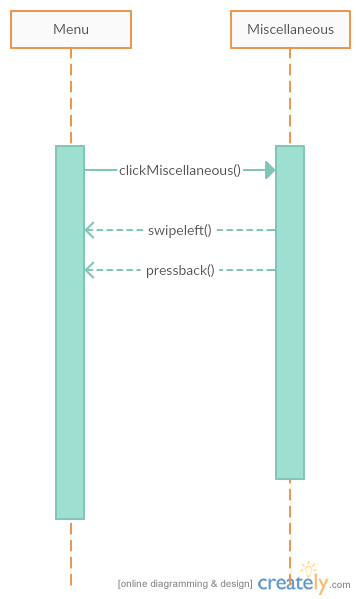
**3.twitter menu**



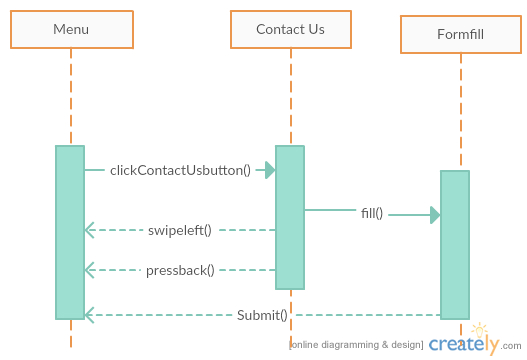
**4.instagram menu**



**5.miscellenous**



**6.contact us**



### **2.3 Structure and relationships**

* Database Engine
  + Existing open source software: MySQL
  + Hosts the backend database which is used for central data storage.
* Class Library
  + Java package which holds all Java classes (and "Java Beans") that both the server and client applications use to represent the data in memory.
  + Provides a uniform representation on both sides of the network connection.
  + Most updates and additions will require only updating this library.
    - Updates to the package *must* be backwards compatible.
  + Client applications can use these objects to represent data, separating it completely from the server-side representation.
* Client Applications
  + Implemented in Java.
  + Contains all presentation logic.
  + Interacts exclusively with the user.
  + Uses the class library package defined above.
  + Communicates with the server application through the server procedure proxy class contained in the class library.

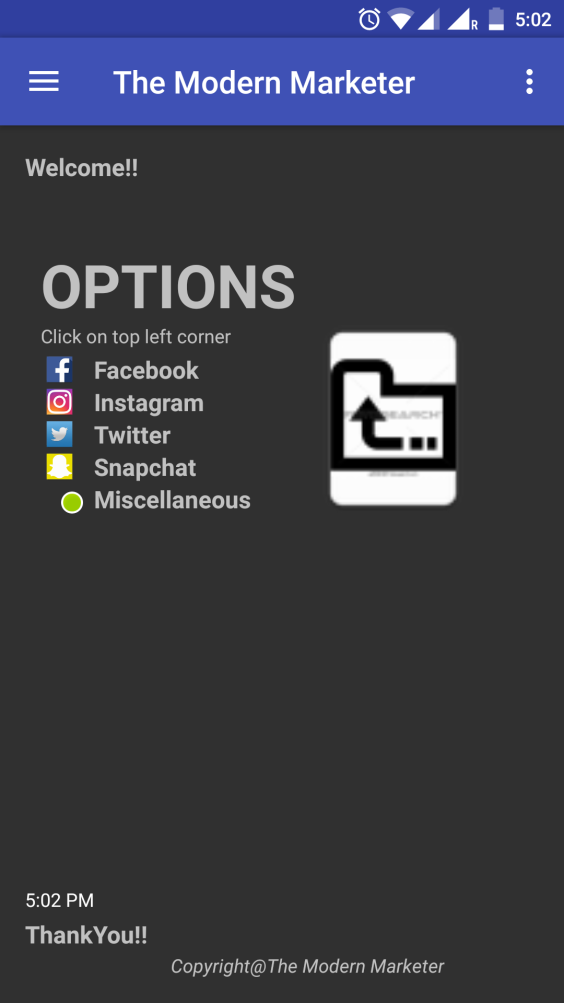
### **2.3 User interface issues**

The user will be able to select the Social Media platform on which he wants to set up his business profile and then follow the detailed steps provided to complete the sign-up, setup his/her business profile, learn to write the perfect post for his business as per the keeping in mind the limitations and the challenges on that respective platform. After this, the user will be able to learn how to increase post engagement, how to increase the footfall, how to gain extra likes and shares on his post etc. The user will be able to switch between the Social Media platforms, and within one social media platform, he/she will be able to switch between the sub-functions of a major social media platform. Apart from these platforms, the user will have access to miscellaneous tips regarding marketting like blogging for traffic creation, youtube, linkedin, medium, live streaming, musical.ly etc. example, *"Blogging is a great way to stay relevant in your niche, as storytelling is really important when you market anything."*

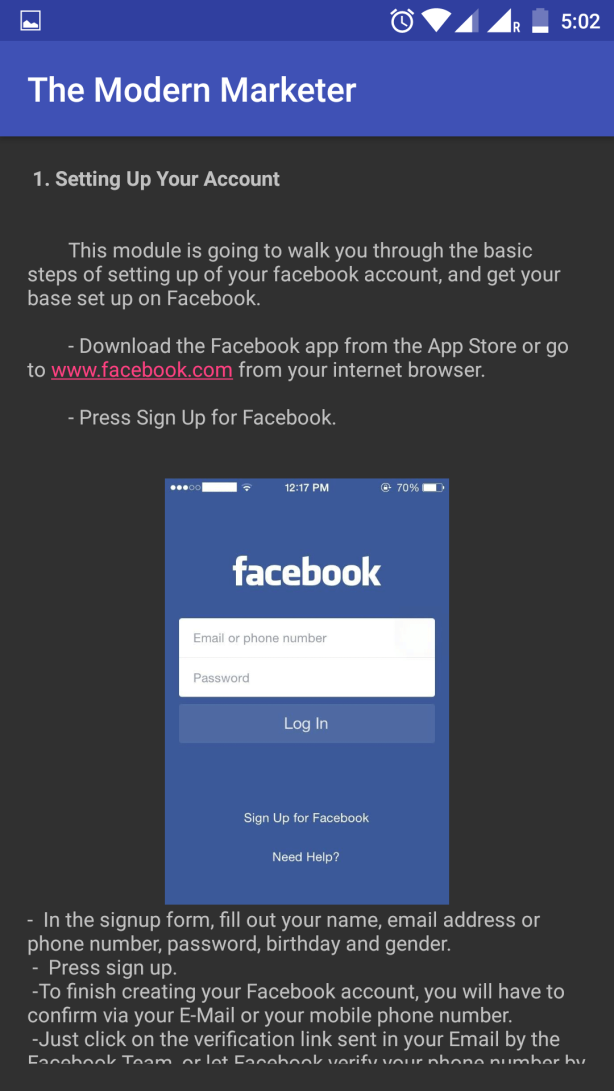
**2.3.1 The main menu**

The main menu will have the option to choose the one of the various social media platforms to work upon, access miscellaneous tips, contact us feature so that the customer can approach us at any point of time.

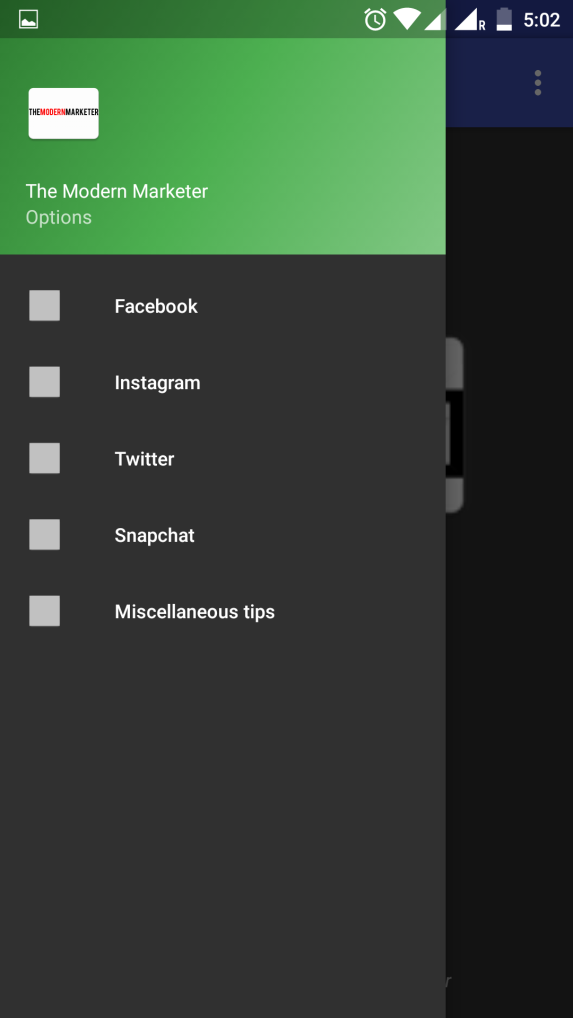
**Main Menu screen-** This first screen will appear at the launch of the application, which will be having the slider on the left consisting of the social media platforms, and miscellaneous tips, the contact us details and the generalised help.



**Social Media platform screen -** This screen will allow the user to enter various subparts within the platform, and to switch between them.



**The subpart screen:**This screen is the last screen, where you can scroll to go through the steps of the respective platform’s profile setup procedure. The user will have to go back to the menu by pressing the back button.



#### **2.3.2 The Settings**

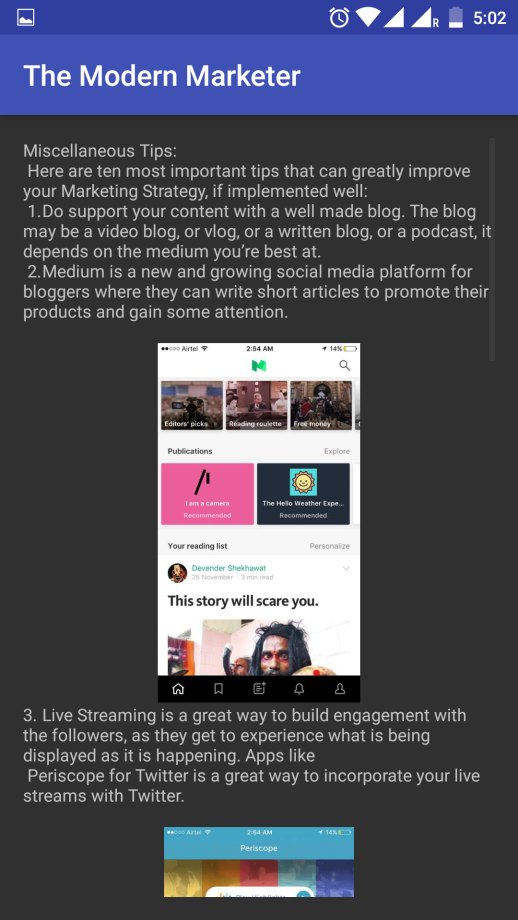
**The Settings screen-** This is a very basic screen which allows the user to set some basic settings like that of notifications and email alerts. The user can go back to the main menu by pressing the back button.

**2.3.3 The Help**

**The help screen:** This is another screen to which the user can refer to when he needs help regarding some app related issues.

**2.3.4 The Miscellaneous Tips**

**The miscellaneous tips screen:** This screen will be encountered by the user from the main menu itself. This contains all the extra tips and tricks and other useful knowledge which helps in increasing more footfalls and talks about other platforms and ways of increasing the reach of the product.



## **3.0 Detailed description of components**

### **3.1 Component Overview**

The following bulleted outline provides a basic overview of the purpose and architecture of our system's major components. The tables in the remainder of section 3 will provide more details on each component in our application:

* [**3.2**](http://www.cs.utah.edu/~jamesj/ayb2005/docs/SDS_v2.htm#3.2) **Database**: We will be using MySQL as our database software. It will contain tables to store Contact information for the “Contact us” menu option.
* [**3.3**](http://www.cs.utah.edu/~jamesj/ayb2005/docs/SDS_v2.htm#3.3) **Class Library**: The Class Library will be a package of classes common to the Server Software and the Client Software. The server software will use these class definitions to know how to store the objects in the database. The client software will use the class definitions in order to ask for and display the correct information about each class type.
* **Application Software**: The Application Software will reside on an android smartphone. Its purpose is to present data to the user as requested. It will help the customer to choose the right class for marketing and getting back to the main menu as per the requirement.

### **3.2 Database**

3.3 Component Template Description

|  |  |
| --- | --- |
| Identification | The unique name for the component and the location of the component in the system. |
| Type | A module, a subprogram, a form, a data file, a control procedure, a class, etc |
| Purpose | Function and performance requirements implemented by the design component, including derived requirements. Derived requirements are not explicitly stated in the SRS - but are implied or adjunct to formally stated SDS requirements. |
| Subordinates | The internal structure of the component, the constituents of the component, and the functional requirements satisfied by each part. |
| Dependencies | How the component’s function and performance relate to other components. How this component is used by other components. The other components that use this component. Interaction details such as timing, interaction conditions (such as order of execution and data sharing), and responsibility for creation, duplication, use, storage, and elimination of components. |
| Interfaces | Detailed description of all external or internal interfaces as well as of any mechanism for communicating through messages, parameters, or common data areas. All error messages and error codes should be identified. All screen formats, interactive messages, and other user interface components (originally defined in the SRS) should be given here. |
| Resources | A complete description of all resources (hardware or software) external to the component but required to carry out its functions. |
| Processing | A full description of the functions presented in the Function subsection. Pseudocode can be used to document algorithms, equations, and logic. |
| Data | For the data internal to the component, describes the representation method, initial values, use, semantics, and format. |

Description to the main menu

|  |  |
| --- | --- |
| Identification | Main menu |
| Type | class/dropdown |
| Purpose | The user screen gives all users to view scroll and toggle through various modules available |
| Function | Takes the user input |
| Subordinates | The screen contains different modules like   * Facebook * Snapchat * Twitter * Instagram * misclleanous   Contact   * About us * Contact us |
| Dependencies | An administrative user must perform database setup functions, adding and modifying the structure of tables so that the server software can store the data as appropriate. |
| Interfaces | All The links are contained on the screen. The screen is designed to be easy to view using the resolution standard on the PDA. |
| Resources | No database. |
| Processing | The only type of processing required is inputting information into the text boxes and navigating to other modules. Each link directs the user to a different screen that corresponds to the link that the user selects. |
| Data | The data in the database will be filled by user when the Submit button is used in the Contact Us module. |

View facebook

|  |  |
| --- | --- |
| Identification | facebook |
| Type | module |
| Purpose | To show and navigate users to the process of how to use facebook for marketing |
| Subordinates | A database read operation |
| Dependencies | Depends on read operation |
| Interfaces | No interfaces |
| Resources | No resources all the data is already stored and no internet connectivity required |
| Processing | Processes the data to be read from the database |
| Data | The data is already stored |

View instagram

|  |  |
| --- | --- |
| Identification | instagram |
| Type | module |
| Purpose | To show and navigate users to the process of how to use instagram for marketing |
| Subordinates | A database read operation |
| Dependencies | Depends on read operation |
| Interfaces | No interfaces |
| Resources | No resources all the data is already stored and no internet connectivity required |
| Processing | Processes the data to be read from the database |
| Data | The data is already stored |

View twitter

|  |  |
| --- | --- |
| Identification | twitter |
| Type | module |
| Purpose | To show and navigate users to the process of how to use twitter for marketing |
| Subordinates | A database read operation |
| Dependencies | Depends on read operation |
| Interfaces | No interfaces |
| Resources | No resources all the data is already stored and no internet connectivity required |
| Processing | Processes the data to be read from the database |
| Data | The data is already stored |

### 

View miscellaneous

|  |  |
| --- | --- |
| Identification | miscellaneous |
| Type | module |
| Purpose | To show and navigate users to the process of how to use other available social media modules for marketing |
| Subordinates | A database read operation |
| Dependencies | Depends on read operation |
| Interfaces | No interfaces |
| Resources | No resources all the data is already stored and no internet connectivity required |
| Processing | Processes the data to be read from the database |
| Data | The data is already stored |

View contact us

|  |  |
| --- | --- |
| Identification | Contact us |
| Type | A subprogram |
| Purpose | For customer satisfaction and innovation |
| Subordinates | Chat box and performance |
| Dependencies | - |
| Interfaces | An independent sub program which lets users contact us directly |
| Resources | No resources |
| Processing |  |
| Data |  |

**4.0 Reuse and relationship to other products**

**Our team is familiar with the concept of not reinventing the wheel when something is already available and usable. Hence, we have made use of a number of components both internal and external to our project. The internal components are modules that we have created and reused throughout our code. Some of the external components are as follows,**

### **4.1 Java and Java tools**

From the beginning we set a goal to make use of any existing code to avoid wasting time duplicating other’s work. We also decided to use open source or freeware solutions wherever possible. Because we are creating software for a defined application, it is possible to use open source technologies in each area. First, we decided to implement our code in Java because it is free and allows us to run on a wide range of operating systems. Next, we chose a development environment that would allow us to edit our Java code. Eclipse was chosen because it is free and has many plug-ins which allow us to use already existing applications. Finally, we chose the open source Eclipse plug-in Jigloo to create the GUI because it allows us to drag and drop frames and edit the generated code all within the same environment.

### **4.2 Database**

With these decisions in place, we needed a database on which to store our records. We chose MySQL because our software requires minimal use of a Database structure for storing Contact information, on the whole, the application requires Database application only for adding and storing Contact Us information.

[Back to Table of Contents](http://www.cs.utah.edu/~jamesj/ayb2005/docs/SDS_v2.htm#Table_of_Contents)

## **5.0 Design decisions and tradeoffs**

### **5.1 Java/.NET Debate**

The first issue we dealt with as a group was whether to develop in Java or .NET. Two of our members wanted to develop in .NET because they are familiar with it and enjoy working with it more. Also, PDA development is easier with .NET. However, while some members of the group do not have .NET experience, all have some Java coding experience. Another important issue we considered in selecting a code was the fact that .NET does not work in all standard operating environments. The range of operating systems that support Java is much larger. Operating systems of interest were: Windows, Unix, Linux, and PocketPC, all of which are supported by Java and not by .NET.

### **5.2 Three Tier Design**

Deciding how to judiciously divide the project between all team members was another design issue. We finally decided on a 3 tier design, which is an application program organized into three major parts, each of which is distributed to different places in a network.

The three parts are:

1. The client application
2. The server application
3. The database and programming related to managing it

A 3-tier application uses the client/server computing model. With three tiers or parts, each part can be developed concurrently by a different team of programmers. Because the programming for a tier can be changed or relocated without affecting the other tiers, the 3-tier model makes it easier for an enterprise or software packager to continually evolve an application as new needs and opportunities arise. Existing applications or critical parts can be permanently or temporarily retained and encapsulated within the new tier of which it becomes a component. This design idea was very appealing to our team, especially for portability purposes. Reference [http://searchdatabase.techtarget.com](http://searchdatabase.techtarget.com/) on 3 Tier design.

### **5.3 Schedule**

As a team we also chose not to include a scheduling feature in our software. There are several existing software products that help you plan for the future but few that let you track things done in the past. We also recognized that we would not have enough time to properly implement a scheduling feature and decided to exclude it from our plan.

### **5.4 Projects/ProjectItems**

We also considered using one class for both projects and sub-tasks. This would allow the user to easily upgrade a sub-task to project status, or to assign the sub-task to another user. The only difference between the two would be the amount of information that the user provided. However, after examining the options, we decided that it will be easier to create two different tasks and provide a means by which the user can "upgrade" a sub-task to project status, at which point they can reassign it if they so desire.

6.0 Pseudo code for the components