**Software Design Description (SDD)**

**CEN5016 Software Engineering**

**Project Statusphere**

1. **Introduction**

This section provides an overview of this entire document.

* 1.1. Project Overview

The team is developing an improved version of the system, Statusphere, for Kristen Wiley of JoinStatus. Statusphere provides select social media *Influencers* with advertising opportunities: in exchange for advertising selected products, an Influencer receives that product for free; the current version of Statusphere uses a simple form-based front-end, hosted on a third-party site, and relies on manual labor to complete most of the services that it aims to provide. An improved front-end, for both the Influencers and for any Administrators, is to be developed, along with a back-end system for storing data and automating several common administrative tasks.

* 1.2. Project Scope

The project entails: building a front-end, both a web portal and a mobile app, for influencers to select advertising *packages*; building a front-end for administrative tasks (such as accepting new influencers and adding new advertising packages); building a back-end for storing both influencer and advertiser package information; building a notification system for facilitating quick, semi-automated communication between Kristen and her users (Influencers); and, developing an automated acceptance system for selecting influencers. The system must include an iOS app; an Android app is not necessary.

* 1.3. Document Preview

This document is intended for the software developers and others with technical expertise; some portions of this document are suited for a general purpose audience, such as section 4, User Interface Design. The document includes sections for Architectural Design, Detailed (Component) Design, Interface Design, and the tracing of requirements. Additional information can be found in the Statusphere Software Requirements Specification document.

* 1.4. References

See <https://github.com/bwj-GitHub/CEN5016-Statusphere/tree/master/Documentation> for Statement of Work, Project Management Plan, and Software Requirement Specification documents.

2. **Architectural Design**  
This section provides an overview and rationale for the program's data and architectural design decisions.

* 2.1. Section Overview

This section discusses the high-level design of the system, Statusphere. It includes: a list of major constraints; a description of the database being used and how it will be deployed, the schema for the database is pictured in **Figure 2.3.1**; the User-Interface is briefly discussed, using a Use-Case Diagram (see **Figure 2.4.1**) as a reference; the three-tier client-server architecture (see **Figure 2.5.1**) of Statusphere is briefly discussed; and, several design alternatives are discussed.

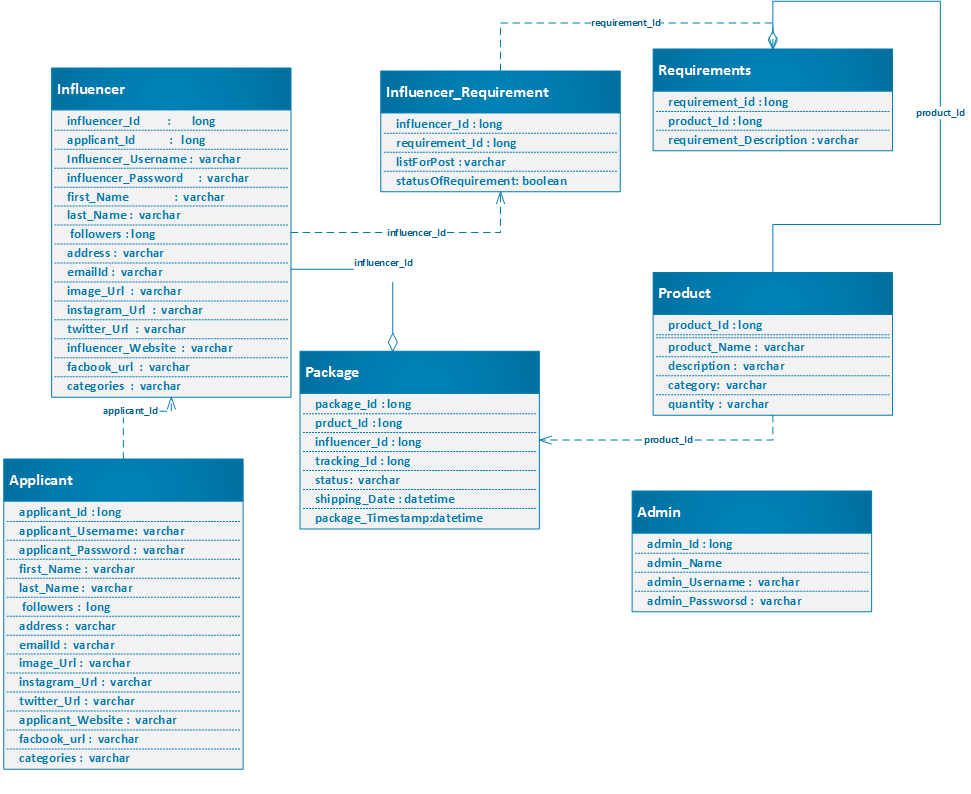
* 2.2. General Constraints

The middleware and back-end components of the system will be ran on an AWS EC2 server. The system will have to interface with the APIs for Facebook, Instagram, Twitter, and YouTube; it must correctly implement their Authentication protocols; and, it must be able to handle JSON objects, which these APIs will use to return requested data. The iOS app must function properly on iPhone versions 6 and 7. The User Interfaces must function properly on the latest versions of Safari, Chrome, and Firefox web browsers.

* 2.3. Data Design

The system will use a single database, consisting of Seven tables: Influencers, Applicants, Admins, Packages, Products, Requirements and Influencer\_Requirements. A product locking mechanism has been applied in the system to ensure concurrent access of same product by multiple users. The moment a product gets checked out by the influencer, it becomes locked into the system. The time-stamp gets captured and the product count gets reduced to enforce locking. The lock remain in place for an active session within which check out has to be completed. For an idle session ( predefined time period), it gets released and product count gets rolled back to the previous value.

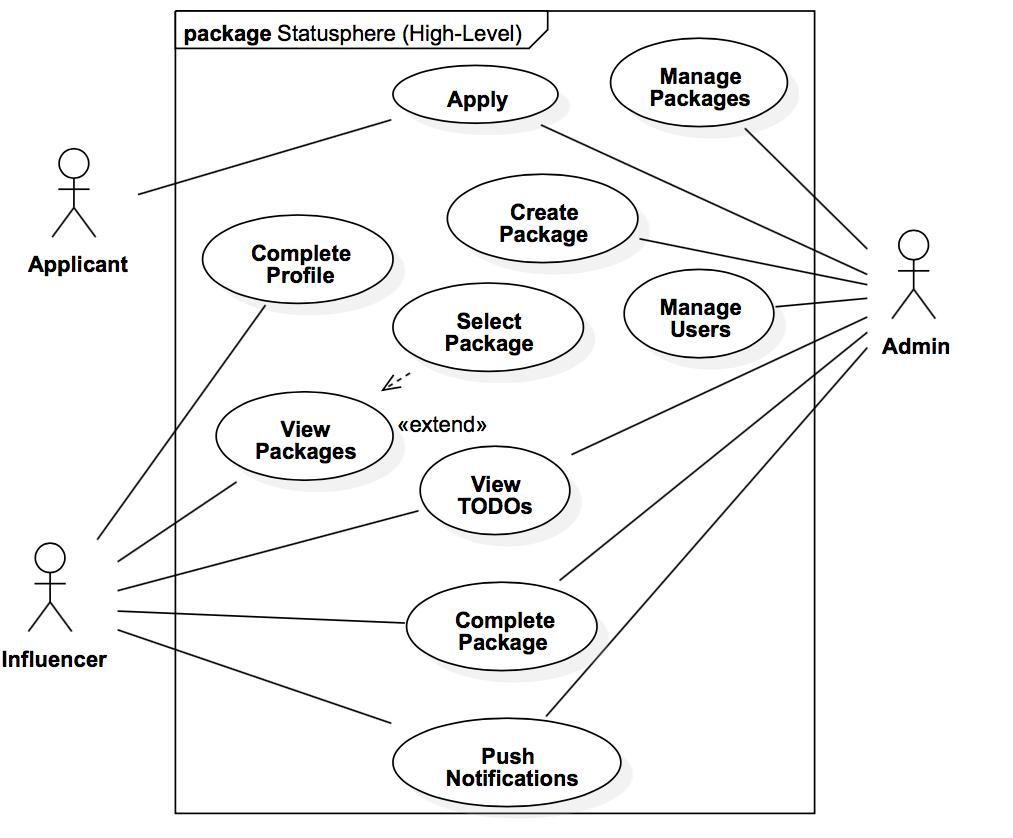
Figure 2.3.1: Entity-Relation Diagram



* 2.4. Interface Design

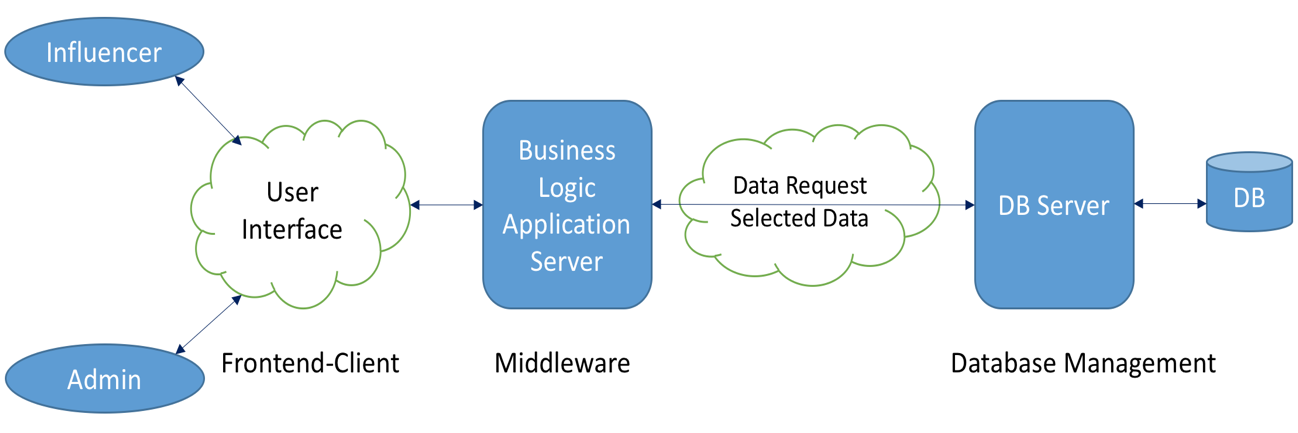
A high-level Use-Case Diagram is shown below in **Figure 2.4.1**. The key roles are played by the Applicant, the Influencer and the Admin. After the applicant applies for the position of an influencer, the admin decides whether the applicant is fit to be one. In addition the admin has to monitor the process which includes creating and managing the packages and managing the users, checking the To-do lists of an influencer and checking if the influencer has completed the requirements of a package and sending out notifications. The influencer after completing his/her profile, has the ability to view and select the packages. In return the influencer has to complete the requirements associated with the packages.

Figure 2.4.1: High-Level Use-Case Diagram



* 2.5. Program Structure  
  The project is based on three-tier client-server architecture (see **Figure 3.1**). The front-end client side has both browser and mobile app support. The middleware consist of business logic which gets executed in application server. The application data is stored in a database which resides in separate database server. The business logic interacts with database server for data transfer. A more detailed architecture can be found in **Figure 3.1**.

Figure 3.1: Statusphere Three-Tier Client-Server Architecture



* 2.6. Alternatives Considered

In this architecture, the database and the application logic are physically (or virtually) separated, this is fairly common design pattern which is meant to improve the security of the system, as it allows the developers to limit the ways in which the database can be modified.

In addition to separating the database from the application logic, a database server is used as an interface between the application logic and the database. This server allows for a standard API to be implemented for interacting with the database and is intended to limit the duplication of code for various queries that will be made from the application server.

The Influencer interfaces include both a web portal and a mobile app. The Ionic Mobile Framework is being used to develop the mobile app using web technologies, as opposed to developing with Swift or Objective C. The primary reason for doing this is that there is no practical way to develop an iOS application without owning a Mac, which the developers do not have sufficient access to; however, using Ionic will allow us to develop much of the App without access to a Mac device – testing will still require a Mac. Using Ionic has the additional benefit of allowing portions of the code used for developing the Web Portal to be re-used in the Mobile app.

3. **Detailed Design (reference Chapter 14)**

This section represents the meat of your document. Be as detailed as time allows.

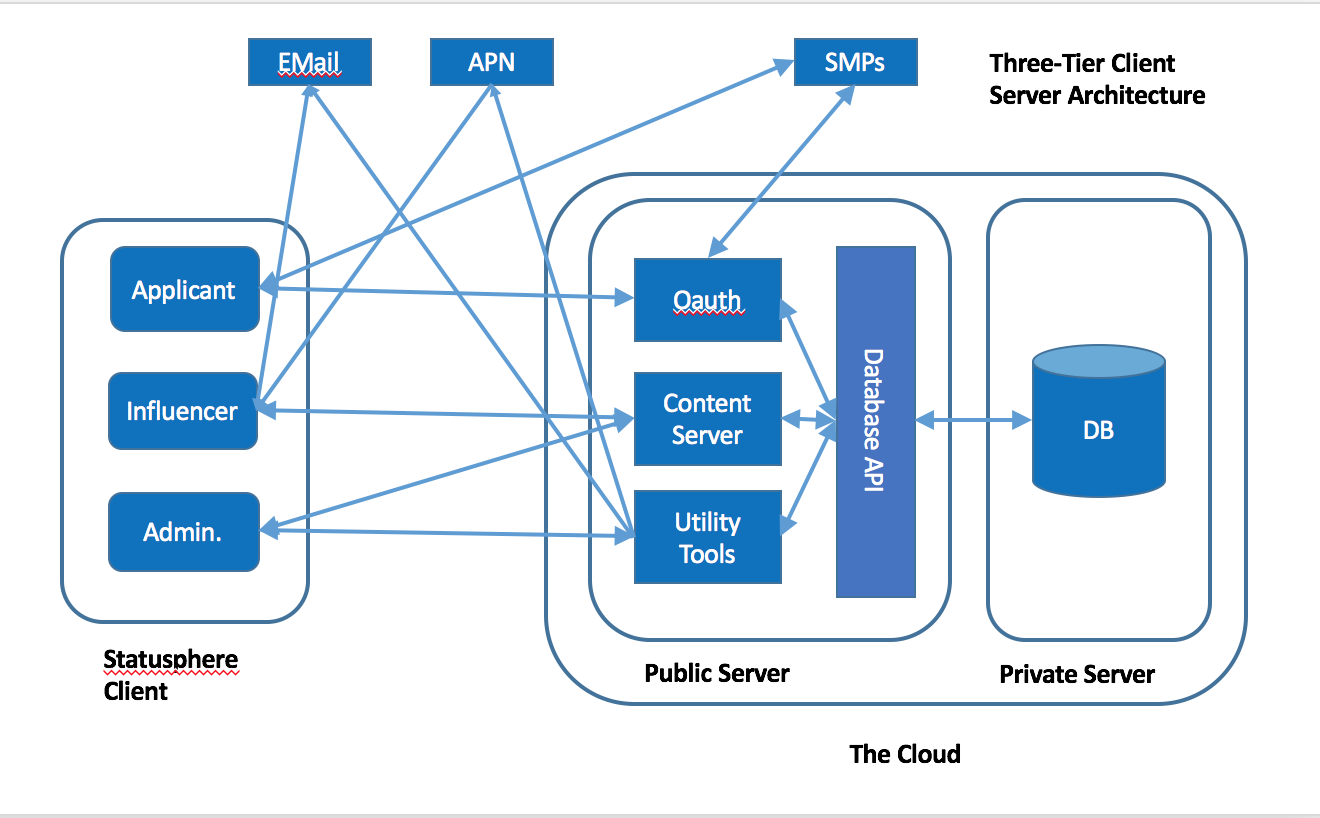
* 3.1. Section Overview

The overall system is essentially a three-tier client server architecture where the actor, here influencer and administrator will interact through web-browser or mobile interface. The business logic in middleware is supposed to process the request from client and respond with the corresponding data. The complete system consists of different subcomponents or modules. The major components includes "OAuth", "place order", "create package", "view package", "manage package" and some utility tools for admin people.

The entire system is to be developed using MEAN web-framework which consist of Mongodb as database, ExpressJS, AngularJS and NodeJS. The frontend consist of web-interface which is planned to be developed using AngularJS framework and a mobile app to be developed using IOS. The backend is going to be developed using NodeJS. To increase security, an database API will be created to interact with the database.

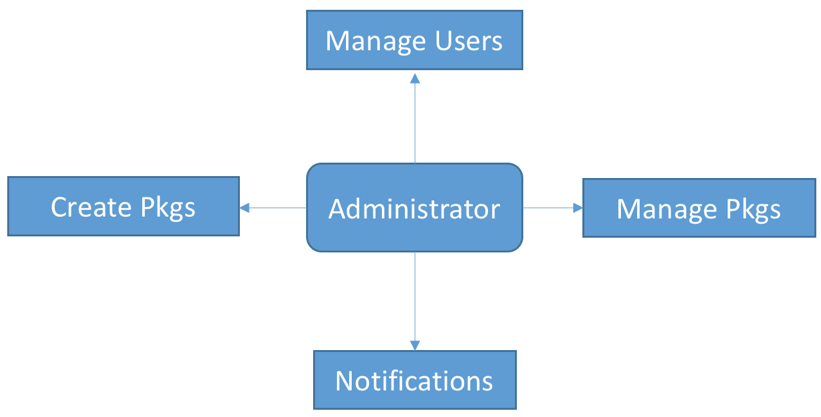
The Statusphere system will have two front-end applications which serve as the client-side applications. One web based written in Angular JS and an iPhone web app using Ionic framework. The various components in the front end include the Applicant class, Influencer class.  
The back end is written in Node JS with a MongoDB database. To increase security, an API will be created to read and write data into the database. The various components in the back end include the Administrator class, OAuth, Administrator Tools and Push Notifications.

**Figure 3.1: Detailed Architecture**



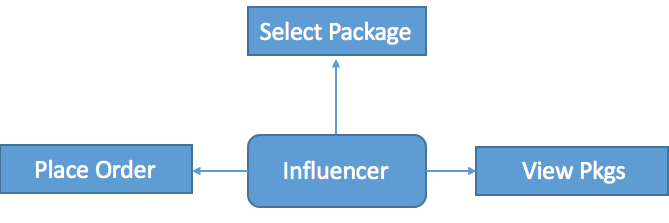
* 3.2. ComponentDetails
  + Administrator:

Administrator is responsible for managing users, creating packages, managing packages and sending notification. The administrator decides whether a applicant is to be accepted and rejected based on some predefined criteria. It creates and manages packages selected by influencer

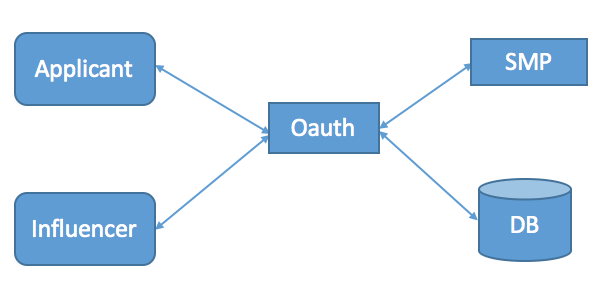


* + Applicant: It sends application to statusphere to become a user of it. The application is verified by administrator to be accepted or rejected.

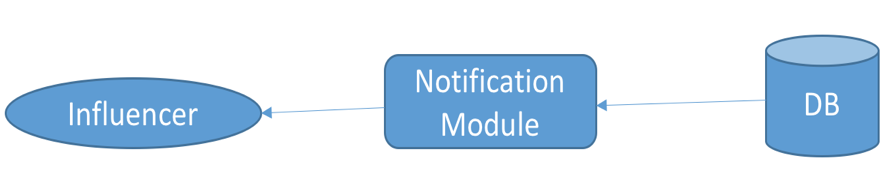
* + Influencer: The accepted applicants are influencer. An influencer can choose the product, place order and view the status of it. The "place order" module is used to place order for selected product. The "view package" module is used to show the status of placed orders.



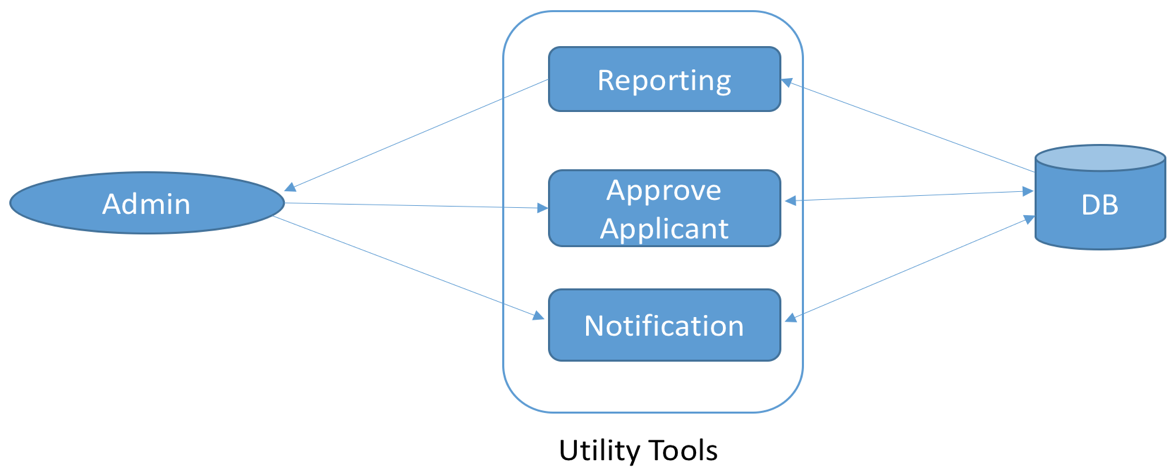
* + OAuth: The user can be authenticated using statusphere login or through social media platform (SMP).



* + Push Notification: The system generated notification messages are sent to the influencer for some incomplete task such as selecting product without placing the order for it. This is only happen when the initiated task is not completed within 24 hours.



* + Utility Tools: The administrative tools or modules are used by admin people only to control the system. It consists of different modules such as reporting, applicant approval, package creation, manage package and notification. The applicant is approved by the admin based on a set of criterion. The reporting module gives an analytical view of the system. The admin views the placed order and initiates the packaging. The "Manage Package" module helps admin to view the status and to make the required changes for the packages. The admin will be able to send a weekly notification to the selective influencers with the updates of new arrivals of products using notification module.



3.2.2. Data Members

Descriptions of each attribute shown in the ERD (**Figure 2.3.1**) are given below:

3.2.2.1 Influencer Table:

*influencer\_id:-*  
Type: long  
Description: Unique identifier for each influencer

*applicant\_id :-*  
Type: long  
Description: Unique identifier for each applicant

*Influencer\_Username :-*  
Type: varchar  
Description: Username for each influencer

*influencer\_Password :-*  
Type: varchar  
Description: The hashed passwords for each username

*first\_Name :-*  
Type: varchar  
Description: First name of each Influencer

*last\_Name :-*  
Type: varchar  
Description: Last name of each Influencer

*followers :-*  
Type: long  
Description: Number of followers of each Influencer

*address :-*  
Type: varchar  
Description: address of each Influencer

*emailID :-*  
Type: varchar  
Description: Email ID of each Influencer

*image\_Url :-*  
Type: varchar  
Description: URL to Influencer image

*instagram\_Url :-*  
Type: varchar  
Description: URL to Influencer’s Instagram account

*twitter\_Url :-*  
Type: varchar  
Description: URL to Influencer’s twitter account

*influencer\_website :-*  
Type: varchar  
Description: URL to Influencer’s website

*facebook\_url :-*  
Type: varchar  
Description: URL to Influencer’s Facebook account

*categories :-*  
Type: varchar  
Description: The domain of the Influencer

3.2.2.2 Influencer\_Requirement Table

*requirement\_Id :*  
Type: long  
Description: The requirement ID of the Influencer

*listForPost*:  
Type: varchar  
Description: Lists the posts for the influencer

*statusOfRequirement:*  
Type: boolean  
Description: status of the requirement

3.2.2.3 Product Table

*product\_Id:*  
Type: long  
Description: The product ID of the product

*requirement\_Description:*  
Type: varchar  
Description: Description of the requirement

*product\_Name*:  
Type: varchar  
Description: Name of the product

*description:*  
Type: varchar  
Description: Description of the product

*category:*  
Type: varchar  
Description: Category of the product  
*quantity:*  
Type: varchar  
Description: quantity of the product

3.2.2.4 Package Table

*package\_Id:*  
Type: long  
Description: The package ID for the package

*tracking\_Id:*  
Type: long  
Description: The tracking ID of the shipment or package

*status:*  
Type: varchar  
Description: Status of the package or shipment

*shipping\_Date:*  
Type: DateTime  
Description: The Date when the package was shipped

*shipping\_Timestamp:*  
Type: DateTime  
Description: The time when the package was shipped

3.2.2.5 Applicant Table

*applicant\_Username:*  
Type: varchar  
Description: The username of the applicant

*applicant\_Password:*  
Type: varchar  
Description: The password of the applicant

3.2.2.6 Admin Table

*admin\_Id:*  
Type: long  
Description: The ID of the admin

*admin\_Name:*  
Type: varchar  
Description: The name of the admin

*admin\_Username:*  
Type: varchar  
Description: The username of the admin

*admin\_Password:*  
Type: varchar  
Description: The password of the admin

**4. User Interface Design**

* 4.1. Section Overview

This section explains the needs of the user and the system’s connection with the influencer, admin and the applicant. The use case diagram gives a brief explanation of the roles and functions of each component of the system-applicant, influencer and admin. First, it talks about the interface design rules which will be used throughout the project. It also explains about the GUI (Graphical user interface) components which will include computer programming to validate forms, establish links and perform desired actions. It also gives a detailed description which is explained step by step by the development of wireframes in the form of simple interactive screens to create a web page and mobile iOS application for influencer and admin portal.

* 4.2. Interface Design Rules  
  The Interface Design Rules for Statusphere are as follows:

**User input validation:**

The system will use HTML basic validations attributes. AngularJS will be used for further validations functionality.

AngularJS will be used to determine whether inputs meet some of the pre-defined rules to ensure that only desired inputs are stored in the database. Detecting and avoiding erroneous inputs before storing in the database, will make system more efficient and secure.

**Rule for error messages, warnings:**

System will use simple but informative error messages to inform user what is wrong with a certain operation. The type of the error will be stated at the top of the error message, enabling user to understand quickly what caused the problem. If fields are not properly filled out when a user presses the submit button, they will be notified, and suggestions will be made to help them fix the mistake.

**Informative feedback:**

Statusphere system will notify users of the upcoming deadlines for the completion of requirements of a package as well as the addition of newly available packages.

**Easy reversal of actions:**

The system will allow users to revise their selections in the same interface where they make changes.

If the influencers submit completed requirements and their links for package is incorrect, he/she can remove the completed requirements and links and re-submit it.

* **4.3. GUI Components**

The initial page in the Statusphere consists of a login or a signup component which enables new user to create an account with Statusphere and existing user to log in with his/her credentials. Once sign up page is clicked, then the user enters his/her information to create an account. The user, once accepted for influencer role, can fill in his/her personal details in the respective text boxes, check boxes and radio buttons.

The package page contains images of respective packages along with their descriptions, requirements and accept/decline checkboxes. The TODO list shows the products which the influencer should advertise in a list component. Each page has few tabs for navigation purposes and few important components like buttons, text boxes, images, check boxes, radio buttons etc. The admin page also contains some GUI components for listing influencer's details, assigning package to the influencer, and tracking their work etc.

The admin login page has text box components to provide their username and password and enter into their portal. The admin then has the option of navigating to her own profile or packages section or influencer's section by selecting the corresponding radio button components. The package section has a list component for packages along with their description. The admin can also check the To-Do's of all influencers from the table component in the portal of respective influencer. The admin can see the details of the applicant by selecting applicant radio button. This page provides applicant's details including their facebook and instagram followers details. The admin can then accept or reject the applicant by selecting the concerned button component. The admin has the right to see his/her own profile and in that page he/she can select respective buttons for editing his/her personal information or adding a new package or editing a package from the list. The notification section in the admin portal enables the admin to send notifications and email to the influencers using the respective buttons.

Visual Studio 2015 is the API which facilitates the design and implementation of these GUI components through AngularJS web technology. Ionic framework is used for designing interactive and responsive pages in the iOS mobile app.

* **4.4. Detailed Description**

This section explains description of the user interface including screen images for the web application and iOS mobile application

**4.4.1 SignUp/Login for web application and iOS app:**

The first time a user visits Statusphere web applicaion and iOS mobile app, the Login/SignUp menu will appear. To become an Influencer first the user has to SignUp using either facebook, Instagram or gmail; or by providing their FullName, Email and Password. So account for the user will be created in the system.

The main component is the login form, where user needs to provide the credentials username and password or login using facebook, instagram or gmail in order to be authorized to use the web application or mobile application.

Figure: 4.4.1.a



Figure: 4.4.1.b

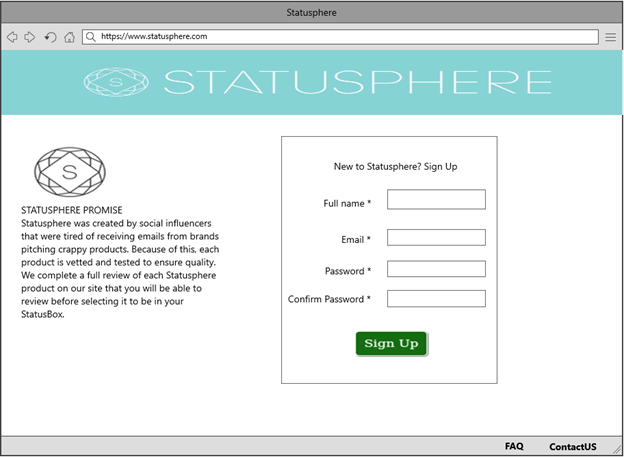
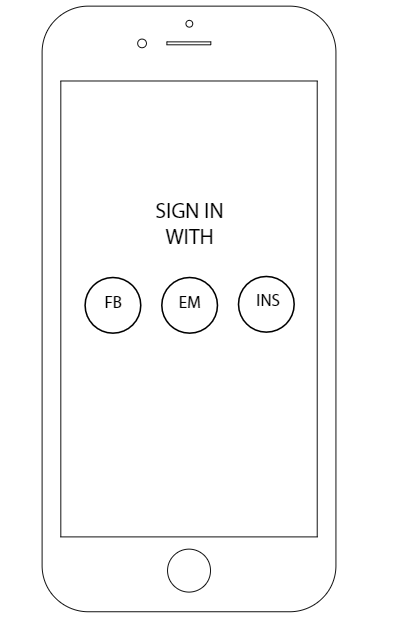


Figure: 4.4.1.c



**4.4.2 Fill Details/Profile:**

After the user logs in to the system, they will be required to fill in additional details for their profile, including their Instagram Account, Facebook and Name, Snapchat Username, No of followers on Instagram, YouTube channel, any blog/website(if any) and categories of their interests .

Figure: 4.4.2.a

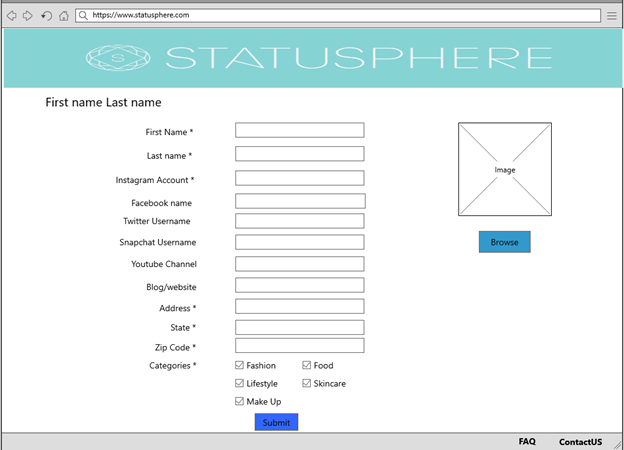
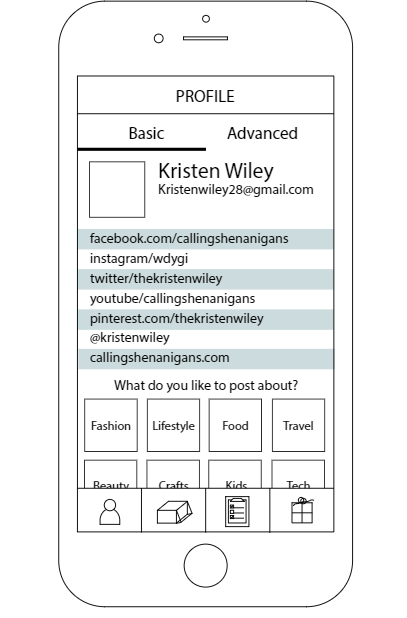


Figure: 4.4.2.b



**4.4.3 View Products:**

The Influencer will be able to see the products of their interests. If the product is already selected by the influencer, the selection has to be disabled, allowing the influencer to select the product again.

Figure: 4.4.3.a

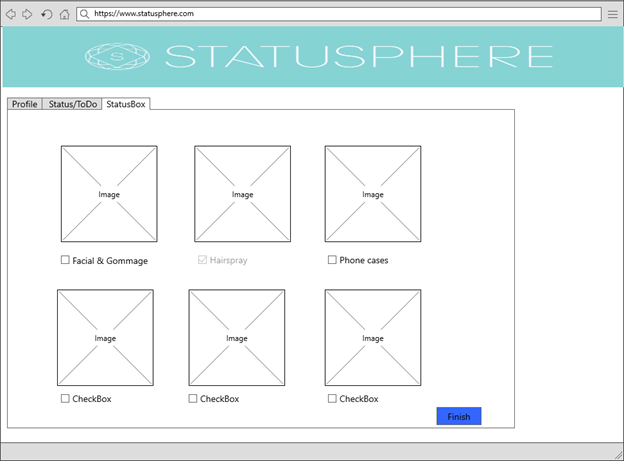
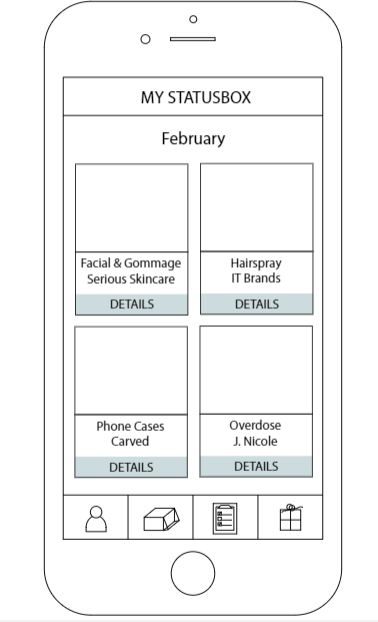


Figure: 4.4.3.b



**4.4.4 Add package to statusbox:**

When the Influencer clicks on product, he/she can see the information about product I.e. description of the product, requirements of the product. If the influencer wants that package, he/she need to select radiobutton "Yes I want this product and I agree to complete the required above actions" and the click on Add to box button. If the influencer does not select Yes I want this product, and clicks on the Add to box button. Error message will get displayed saying select Yes I want this product.

Figure: 4.4.4.a

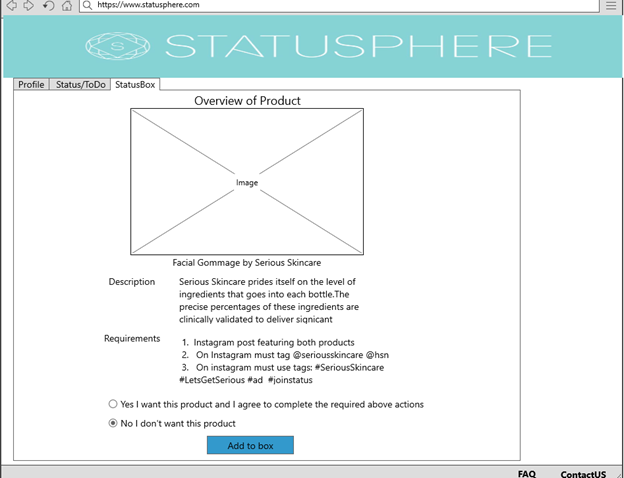


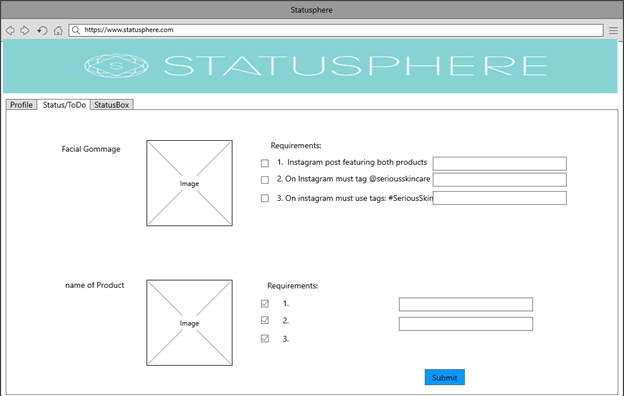
Figure: 4.4.4.b



**4.4.5 Status of selected products/ToDo:**

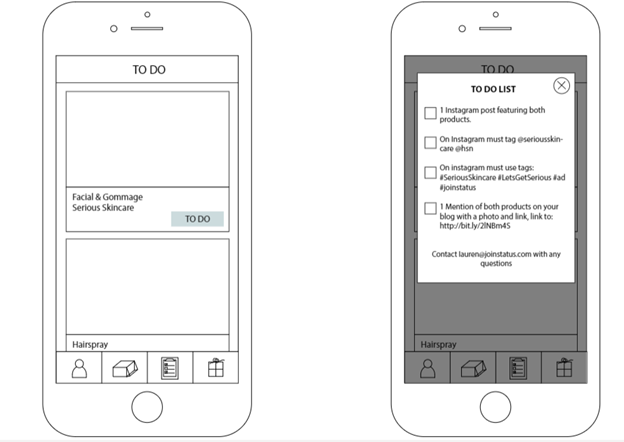
The Influencer can see the list and the status of the selected products and can submit the links for Instagram post for the requirements he/she has completed.

Figure: 4.4.5.a



For mobile app, the Influencer can see the list of To Do products. After the user clicks on TODO button for product, a pop up will be open to show the requirements for product. The user can select the checkbox for the completed requirement for the product.

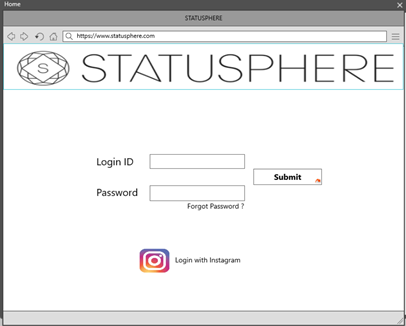
Figure: 4.4.5.b Figure: 4.4.5.c



**Admin portal descriptions:**

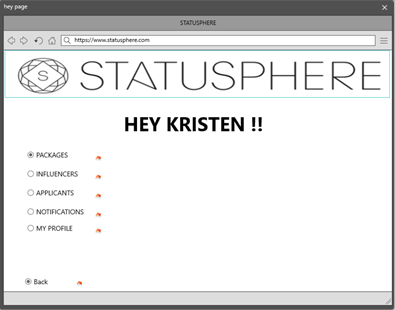
4.4.6. The first webpage is the login page. The admin logins into her account using their login ID and password.

Figure: 4.4.6



4.4.7. After logging in, the admin gets an option to choose any of the field which they are interested in like packages, influencers, applicants, notifications and my profile.

Figure: 4.4.7



4.4.8. When admin selects the 'packages' option, the different packages that are available are displayed. When the particular package is selected, the complete details of the package is displayed.

Figure: 4.4.8.a

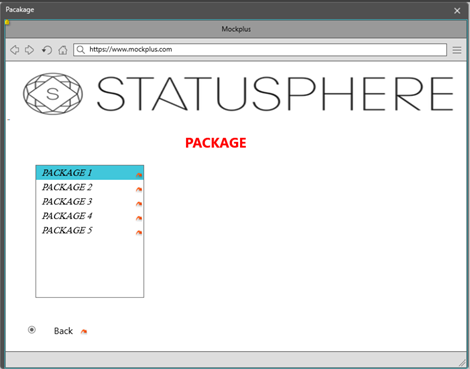
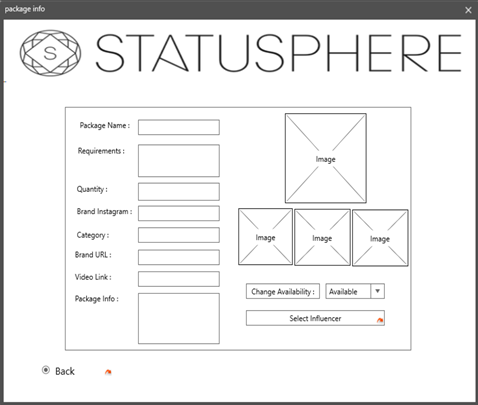


Figure: 4.4.8.b



4.4.9. The admin has the list of the influencers along with the package status. The admin has the access to the complete information of an influencer as well as the access to their TO-Do lists for uncompleted packages of the influencers.

Figure: 4.4.9.a

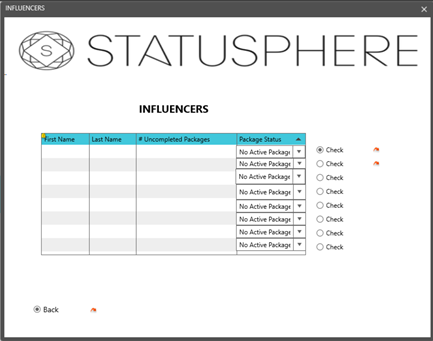


Figure: 4.4.9.b

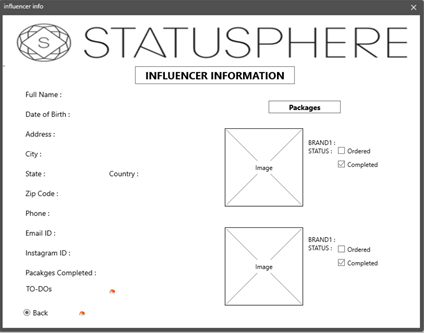
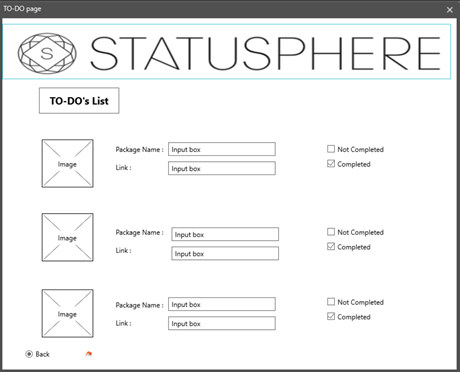


Figure: 4.4.9.c



4.4.10. The admin gets complete list of the applicants along with the #Instagram followers. Here the admin has an option to either accept or reject a particular applicant depending on their popularity on social media platforms.

Figure: 4.4.10.a

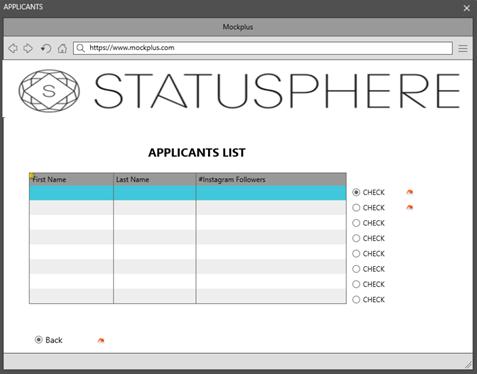
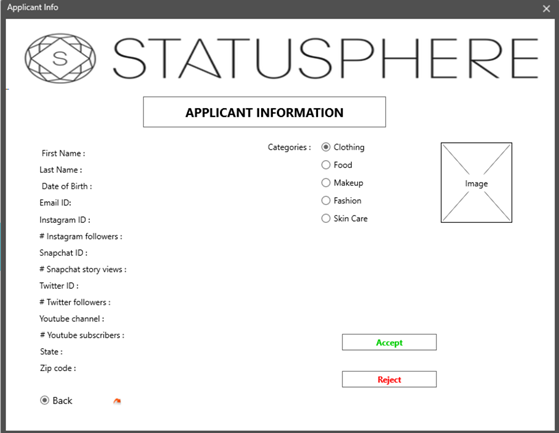
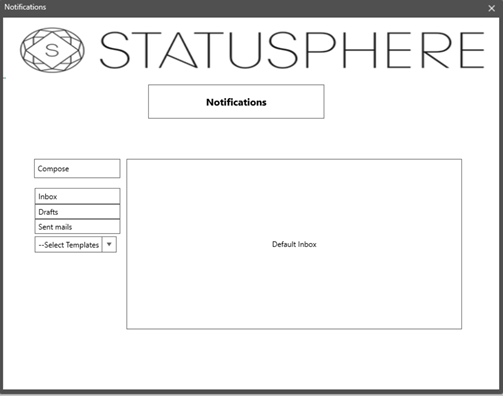


Figure: 4.4.10.b



4.4.11. In the notification page, the admin has the options to view the inbox, compose, draft or send mails to the influencers. The Admin can compose mails from the templates that are available on the notifications page. Templates are generated while designing the notification page.

Figure: 4.4.11



4.4.12. Admin has a profile page which allows her edit her personal information and also manage the creation and edition of packages that are updated to the influencers page which is viewed by the influencers.

Figure: 4.4.12.a

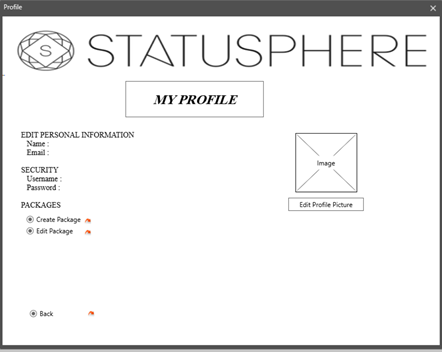


Figure: 4.4.12.b

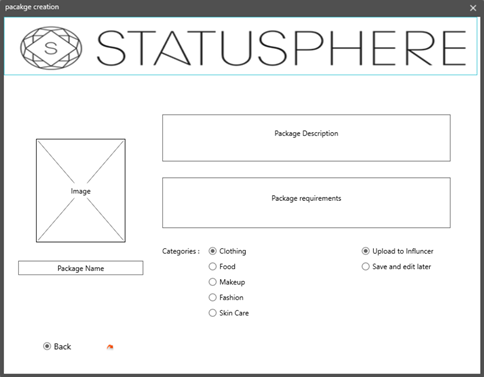
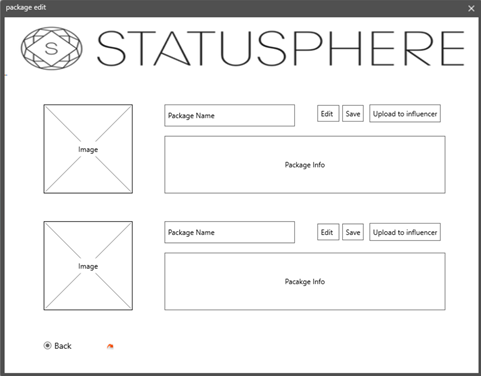


Figure: 4.4.12.c



**5. Requirements Traceability Matrix**

Below is a requirements traceability matrix mapping functional and interface requirements to: 1) the Architecture Component(s) in which they are fulfilled; and 2), the detailed component or interface design component(s) that they are fulfilled in. Test Case Ref. and Status will be filled-in as testing progresses.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Req ID | Requirement Description | Architecture Reference | Design Reference (component / module) | Test Case Ref. | Status |
| FR001 | Influencer application | 3.1.Applicant | 4.4.1.b | TBD |  |
| FR002 | Applicant Acceptance | 3.1.Admin | 4.4.10.b |  |  |
| FR003 | Applicant Rejection | 3.1.Admin | 4.4.10.b |  |  |
| FR004 | Profile Completion | 3.1.Influencer | 4.4.2.a |  |  |
| FR005 | Basic Influencer Login | 3.1.Influencer | 4.4.1.a |  |  |
| FR006 | Login with SMP | 3.1.Influencer, 3.1.OAuth | 4.4.1.[a,c] |  |  |
| FR007 | Influencer Password Reset | 3.1.Influencer | TBD |  |  |
| FR008 | View Packages | 3.1.Influencer, 3.1.ContentServer | 4.4.3.[a,b] |  |  |
| FR009 | Place User on Probation | 3.1.Admin | TBD |  |  |
| FR010 | Accept Package | 3.1.Influencer | 4.4.4.[a,b] |  |  |
| FR011 | Add Package to TODOs | 3.1.ContentServer, 3.1.DB | 2.3.1.Package |  |  |
| FR012 | View TODOs (Influencer) | 3.1.Influencer | 4.4.5.[a,b,c] |  |  |
| FR013 | Prove Package Completion | 3.1.Influencer | 4.4.5.[a,c] |  |  |
| FR014 | Push Notifications | 3.1.Admin, 3.1.UtilityTools | 4.4.7 |  |  |
| FR017 | 24 Hours Remaining Notification | 3.1.UtilityTools | TBD |  |  |
| FR018 | Admin Login | 3.1.Admin | 4.4.6 |  |  |
| FR019 | View Influencers Data | 3.1.Admin, 3.1.ContentServer | 4.4.7 |  |  |
| FR020 | View Packages Data | 3.1.Admin, 3.1.ContentServer | 4.4.7, 4.4.8.a |  |  |
| FR021 | View Influencer-Package Data | 3.1.Admin, 3.1.ContentServer | 4.4.7\*, 4.4.8.a\* |  |  |
| FR022 | Filter Data Tables | 3.1.Admin, 3.1.ContentServer | TBD |  |  |
| FR023 | Remove User | 3.1.Admin, 3.1.Database | TBD |  |  |
| FR024 | View Influencer's TODOs (as Admin) | 3.1.Admin | 4.4.9.[a,b,c] |  |  |
| FR025 | Mark Package as Completed | 3.1.Admin | 4.4.9.[b,c] |  |  |
| FR026 | Remove Completed Package from TODOs | 3.1.UtilityTools, 3.1.DB | 2.3.1.Package |  |  |
| FR027 | Draft New Package | 3.1.Admin, 3.1.DB | 4.4.8.b |  |  |
| FR028 | Push Drafted Packages | 3.1.Admin, 3.1.DB | 4.4.8.b\*\* |  |  |
| FR029 | Look-up No. of Followers | 3.1.Admin, 3.1.UtilityTools | TBD |  |  |
| FR030 | Look-up No. of Likes | 3.1.Admin, 3.1.UtilityTools | TBD |  |  |
| FR031 | Calculate Trimmed Avg. Likes per Post | 3.1.Admin, 3.1.UtilityTools | TBD |  |  |
|  |  |  |  |  |  |
| IR001 | Applicant account creation | 3.1 Applicant | Figure: 4.4.1.b |  |  |
| IR001b | Sign up using social media credentials | 3.1 Applicant  3.1 SMP | Figure: 4.4.1.a |  |  |
| IR002 | Administrator's notification of acceptance/rejection | 3.1 Admin  3.1 Email | Figure: 4.4.11 |  |  |
| IR003 | Influencer profile form | 3.1 Influencer | Figure: 4.4.2.a |  |  |
| IR004 | Validation of user's login credentials | 3.1 OAuth |  |  |  |
| IR005 | Selection of packages based on User's interests | 3.1 Influencer | Figure: 4.4.3.a |  |  |
| IR006 | User's task for the accepted product | 3.1 Admin  3.1 Influencer | Figure: 4.4.5.a |  |  |
| IR007 | Access to user-package details by admin | 3.1 Admin | Figure: 4.4.9.a |  |  |
| IR008 | Verification of a completed package by admin | 3.1 Admin | Figure:4.4.9.b,  Figure: 4.4.9.c |  |  |
| IR009 | Assignment of a new package by admin based on user's previous performance | 3.1 Admin  3.1 Database |  |  |  |

\* This requirement can be filled with a sequence of actions across multiple pages; however, in the SRS, it was intended for a single page to fulfill this requirement.

\*\* The admin is able to push a single drafted package using this interface; however, it was implied in the SRS that the admin could push all drafted packages simultaneously – this will be made explicit in future versions of SRS, and the design will be changed accordingly.

6. **Conclusion**

Some aspects of the design presented in this document require further review or have not yet been finalized, various necessary changes to the Software Requirements Specification have also been elucidated to the developers; these changes will be indicated in future versions of both the SRS and this document. The Requirements Traceability Matrix in section 5, once updated to reflect requirements changes, will be used for monitoring the progress of testing and tracking the completion of modules.

7. **Appendices**   
All diagrams are located in (or near) the sections that they are first referenced.