Software Design Specification

for

Q-Taskers

Prepared by :

<Karan Inder Singh(U101116FCS057)

Rishabh Gaur(U101116FCS100)

Rahul Saha(U101116FCS241)

Rajat Srivastav(U101116FCS098)

> <Software Engineering CS-301>

Software Design Specification

Table of Contents

SOFTWARE DESIGN SPECIFICATION

The Software Design Specification Outline

1. Introduction

The Software Design Document is a document to provide documentation which will be used to aid in software development by providing the details for how the software should be built. Within the Software Design Document are narrative and graphical documentation of the software design for the project including use case models, sequence diagrams, state diagram, class diagram and other supporting requirement information.

1.1 Purpose of this document

This document will define the design of the Business Application. It contains specific information about the expected input, output, classes, and functions. The interaction between the classes to meet the desired requirements are outlined in detailed figures at the end of the document.

1.2 Scope of the development project

We describe what features are in the scope of the software and what are not in the scope of the software to be developed.

In Scope:

a. Application for the Business which have proper Hierarchy

b. Service Provider can retrieve the information about the User.

c. Ease in Maintaining a Record.

Out of Scope:

a. Errors Due to Location Services (Google Maps API)

1.3 Definitions, acronyms, and abbreviations IEEE: Institute of Electrical and Electronics Engineers

IEEE: Institute of Electrical and Electronics Engineers

SDS: Software Design Specification

1.4 Overview of document

This SDS is divided into 5 sections with various sub-sections. The sections of the Software Design Document are:

1. **Introduction:** describes about the document, purpose, scope of development project definitions and abbreviations used in the document.

2. **Logical Architecture**: describes Logical Architecture Description and Components.

3. **Execution Architecture:** defines the runtime environment, processes, deployment view.

4. **Design Decisions and Trade-offs:** describes the decisions taken along with the reason as to why they were chosen over other alternatives.

5. **Appendices:** describes subsidiary matter if any.

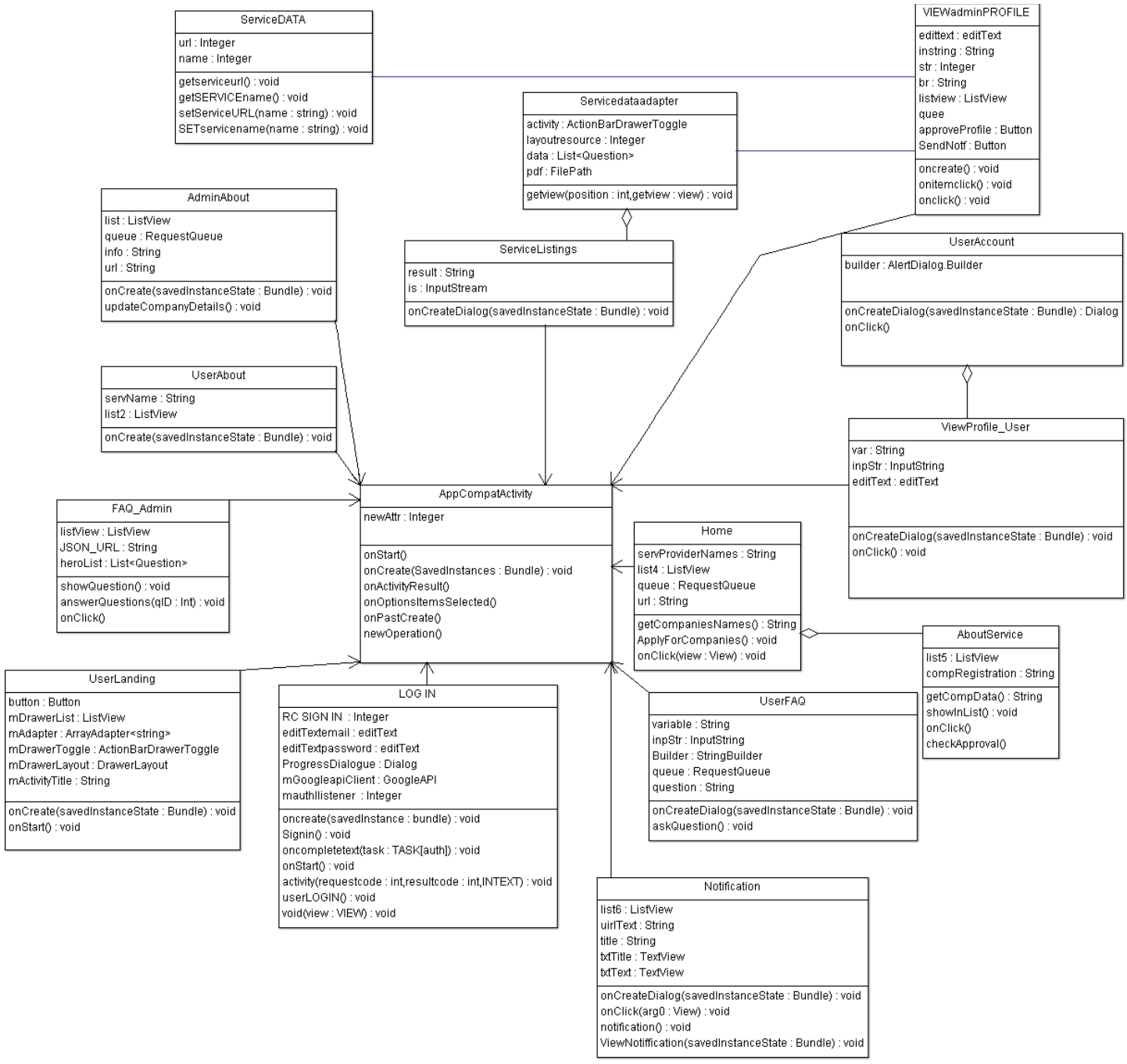
2.1 Logical Architecture Description

2.1.1 Class Diagram explanation:

Most of the classes extends AppCompatActivity class which is being shown by association linkage. It is being shown by black-coloured diamond. The classes which extends AppCompatActivity are: UserAbout, AdminAbout, ServiceDataAdapter, SerivceListings, FAQ_Admin, UserLanding, Login, Notification, UserFAQ, Home, ViewAdminProfile, AboutService, ServiceData, AdminAbout, UserAbout, ViewProfile_User, UserAccount.

3. Logical Architecture (Class Diagram, Sequence Diagram, State Diagram)

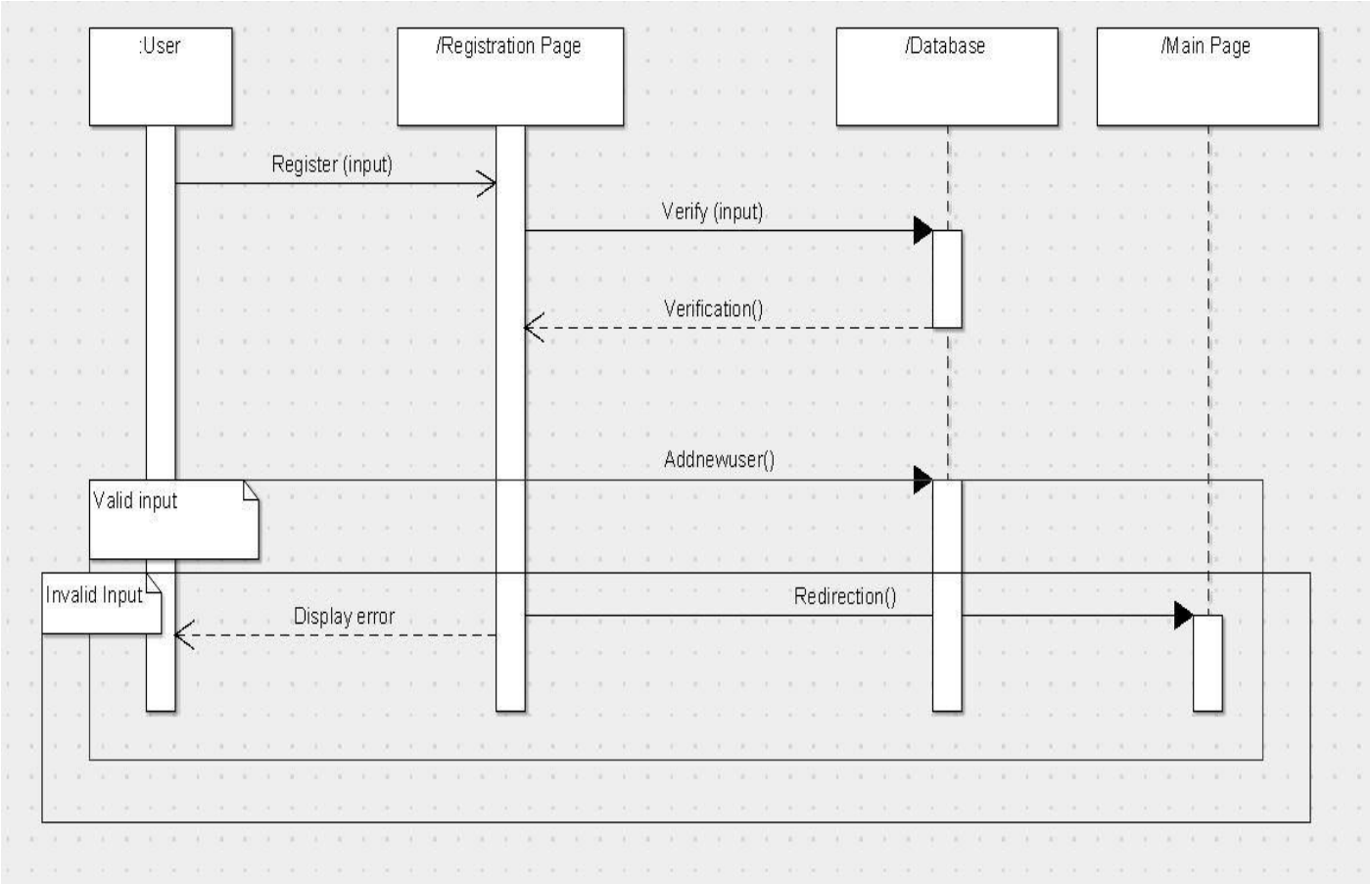
Class Diagram:



Sequence Diagram:

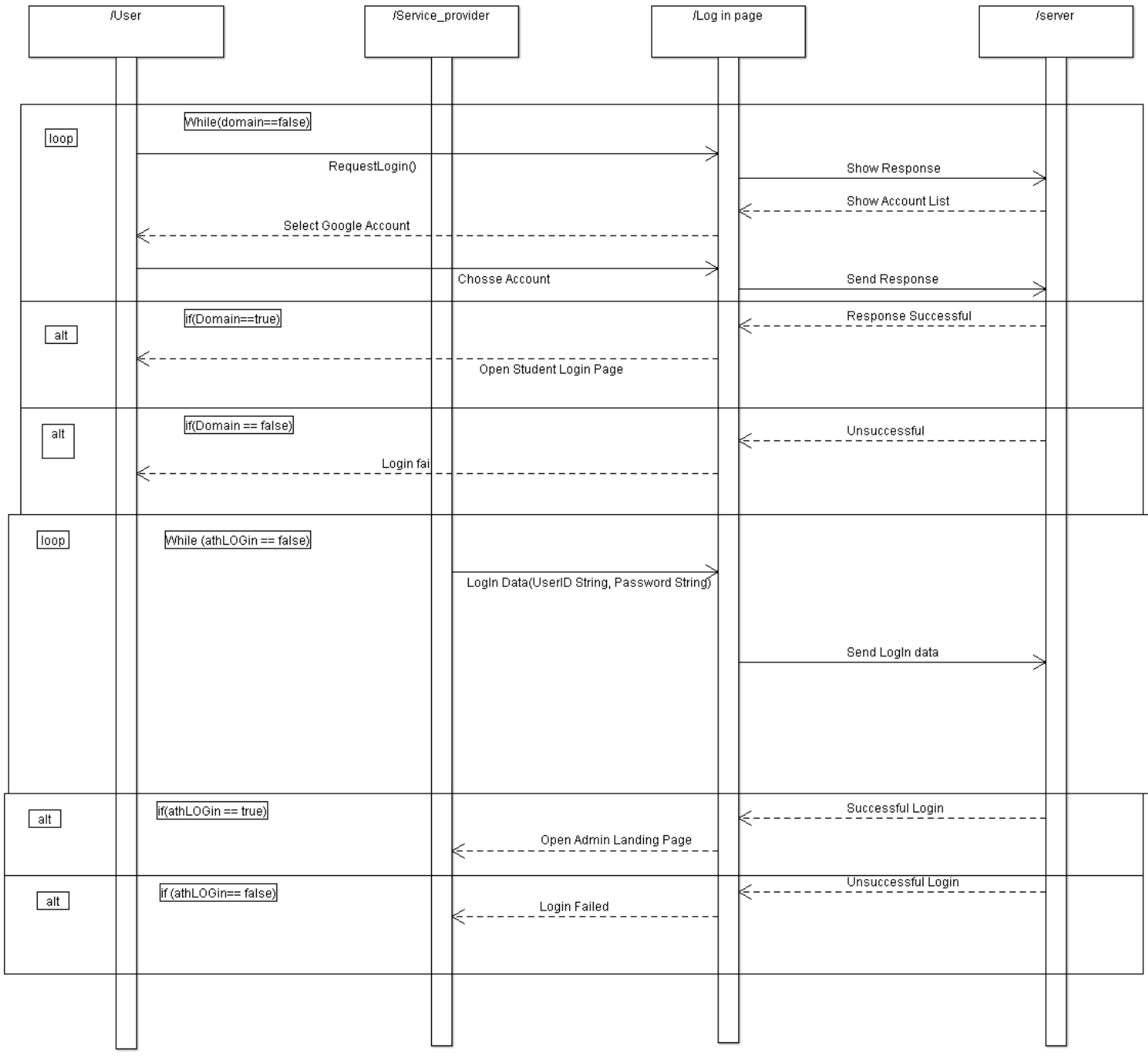
(Service Provider, User, Admin)

SignUp


:User
/Registration Page
/Database
/Main Page
Register (input)
Verify (input)
Verification()
Addnewuser()
Valid input
Invalid Input
Display error
Redirection()

User:-

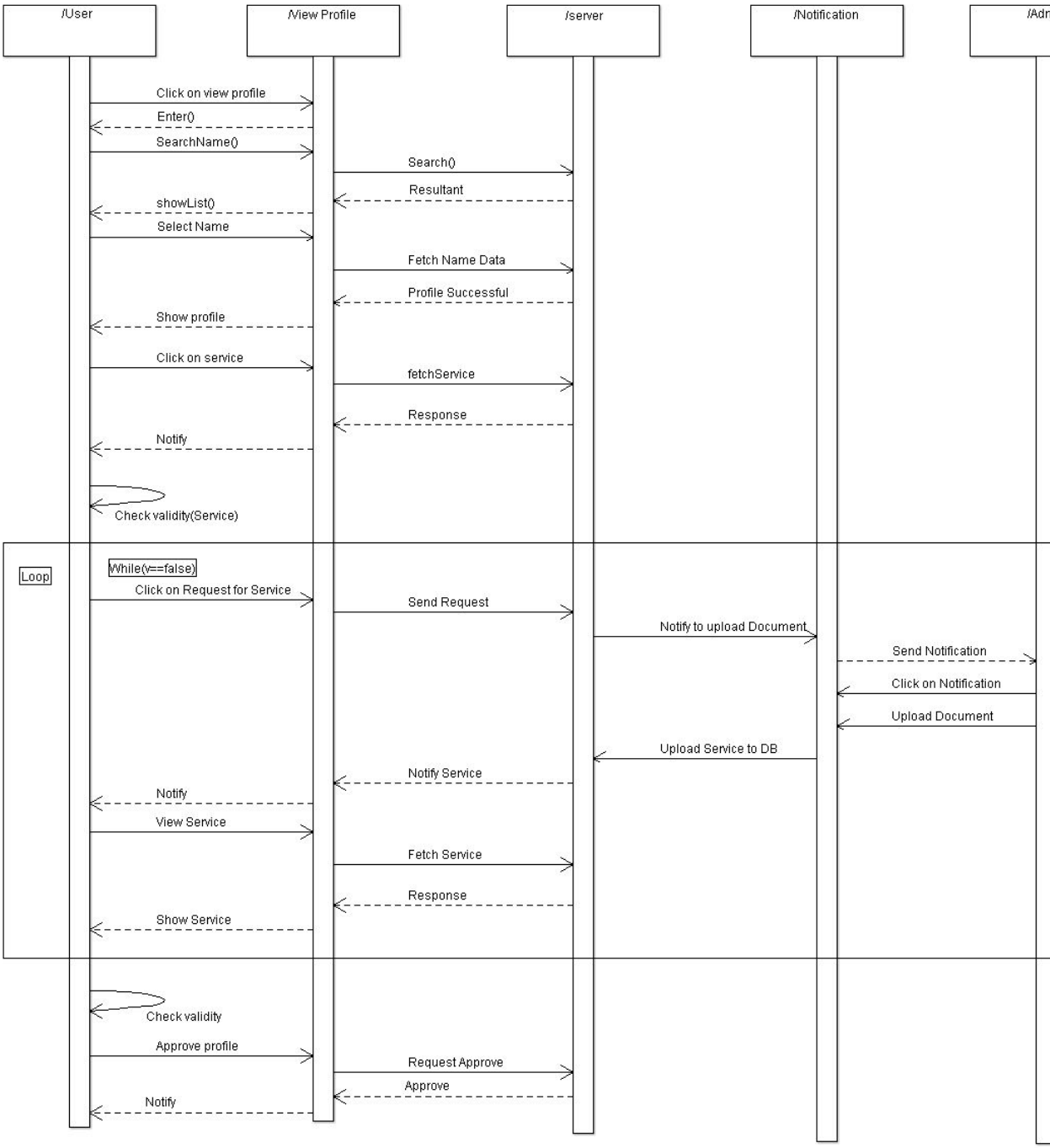
User Login


/User
/Service_provider
/Log in page
/server
loop
While(domain==false)
RequestLogin()
Show Response
Show Account List
Select Google Account
Chosse Account
Send Response
alt
if(Domain==true)
Response Successful
Open Student Login Page
alt
if(Domain == false)
Unsuccessful
Login fai
loop
While (athLOGin == false)
LogIn Data(UserID String, Password String)
Send LogIn data
alt
if(athLOGin == true)
Successful Login
Open Admin Landing Page
alt
if (athLOGin== false)
Unsuccessful Login
Login Failed

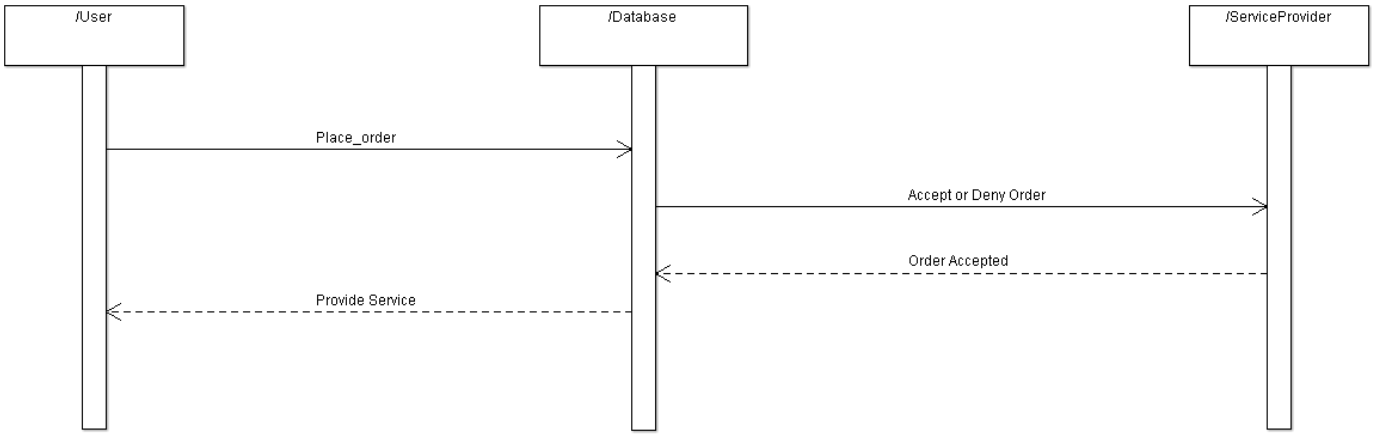
User Display Profile Details

&

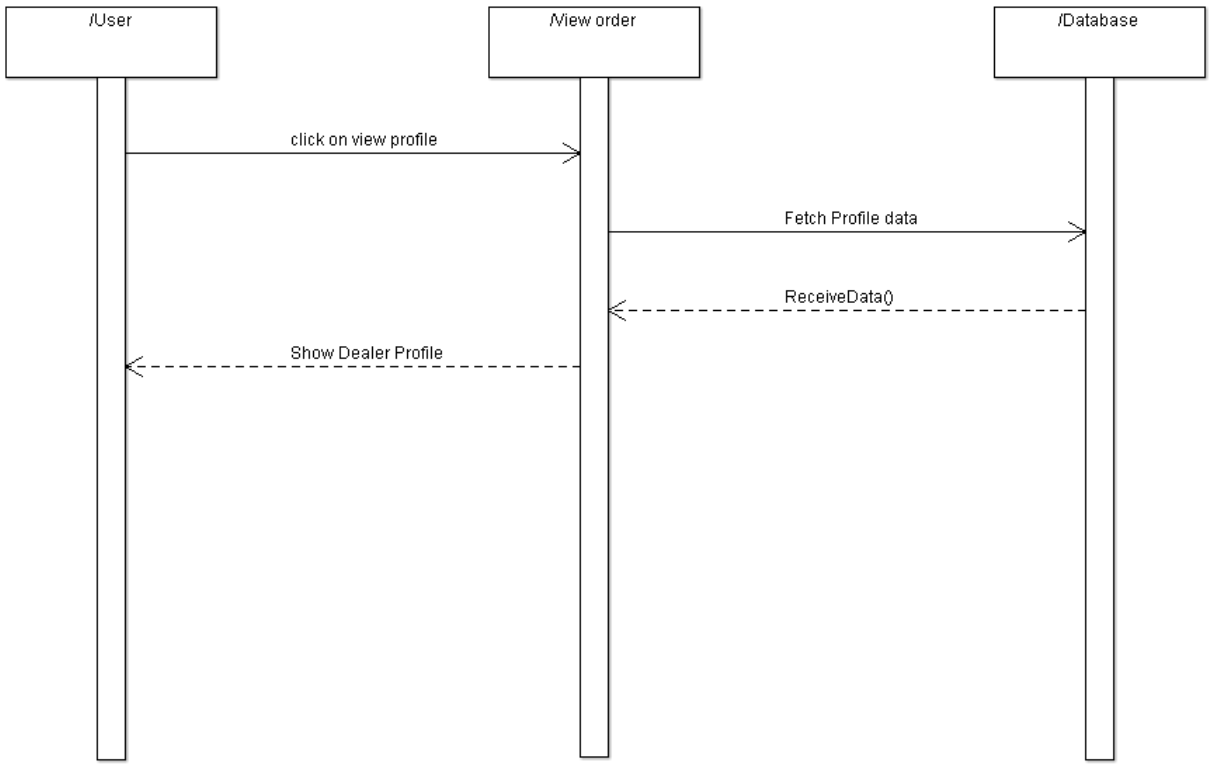
Update Details



User Place Order



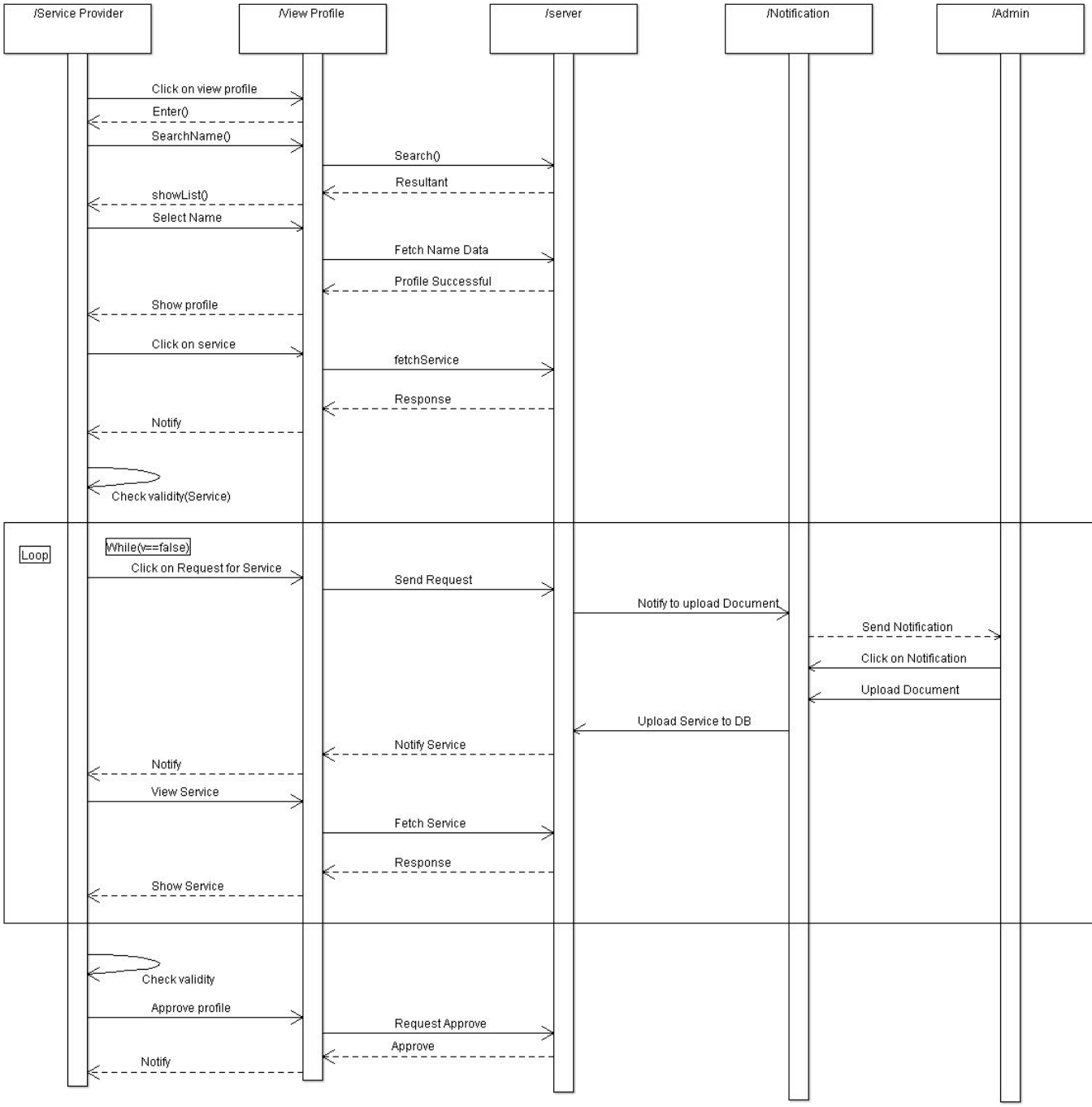
User View Order



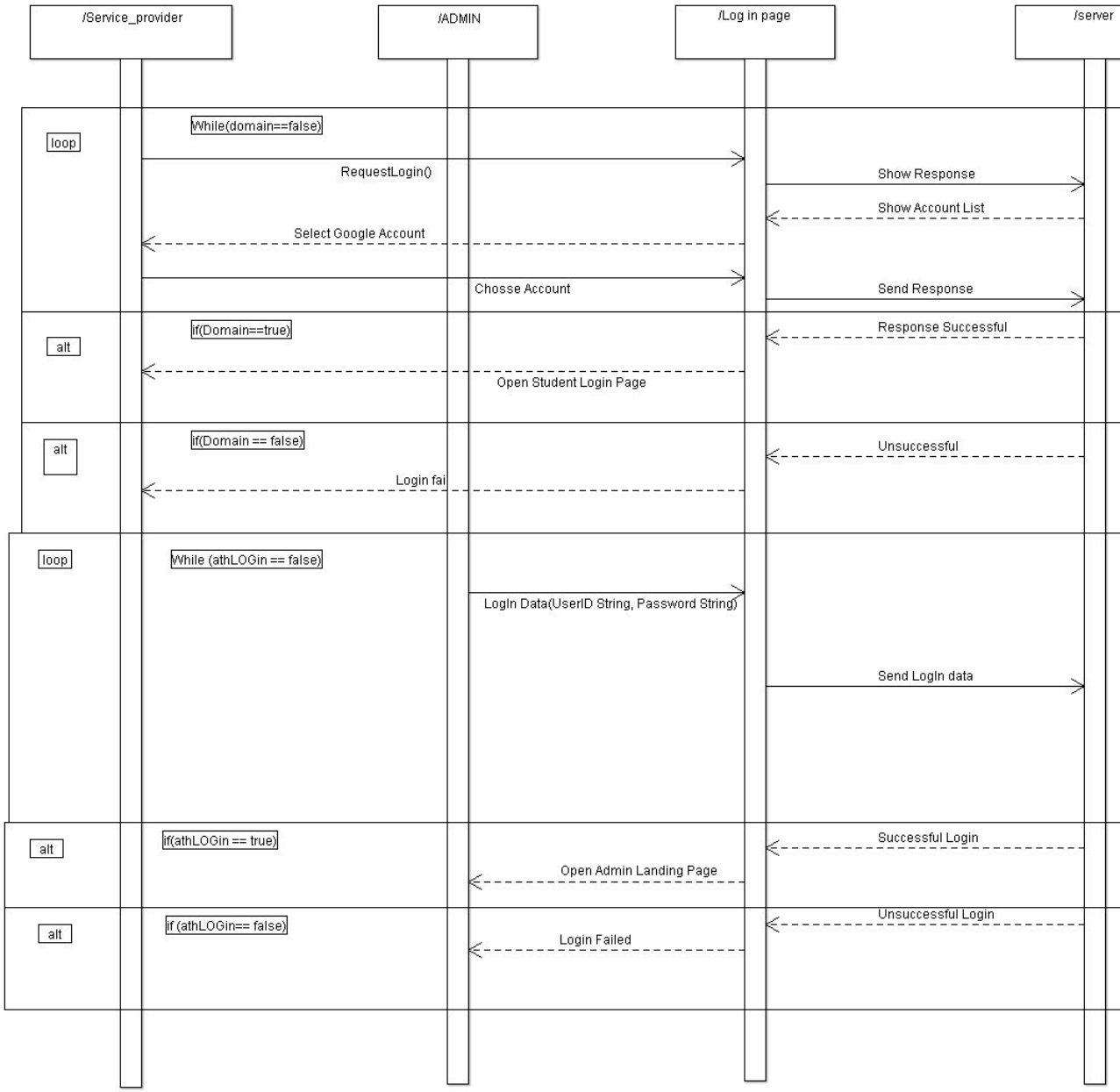
Service Provider:-

Service Provider Display Profile Details &

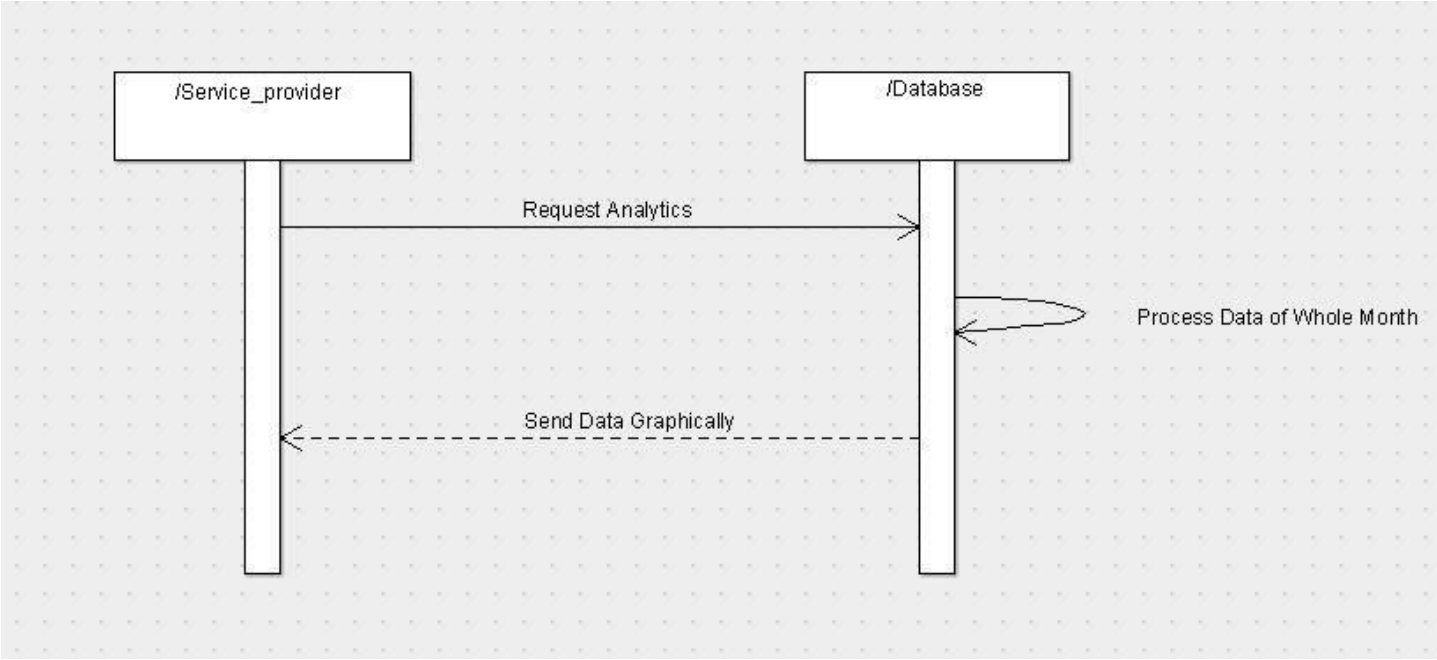
Update Details



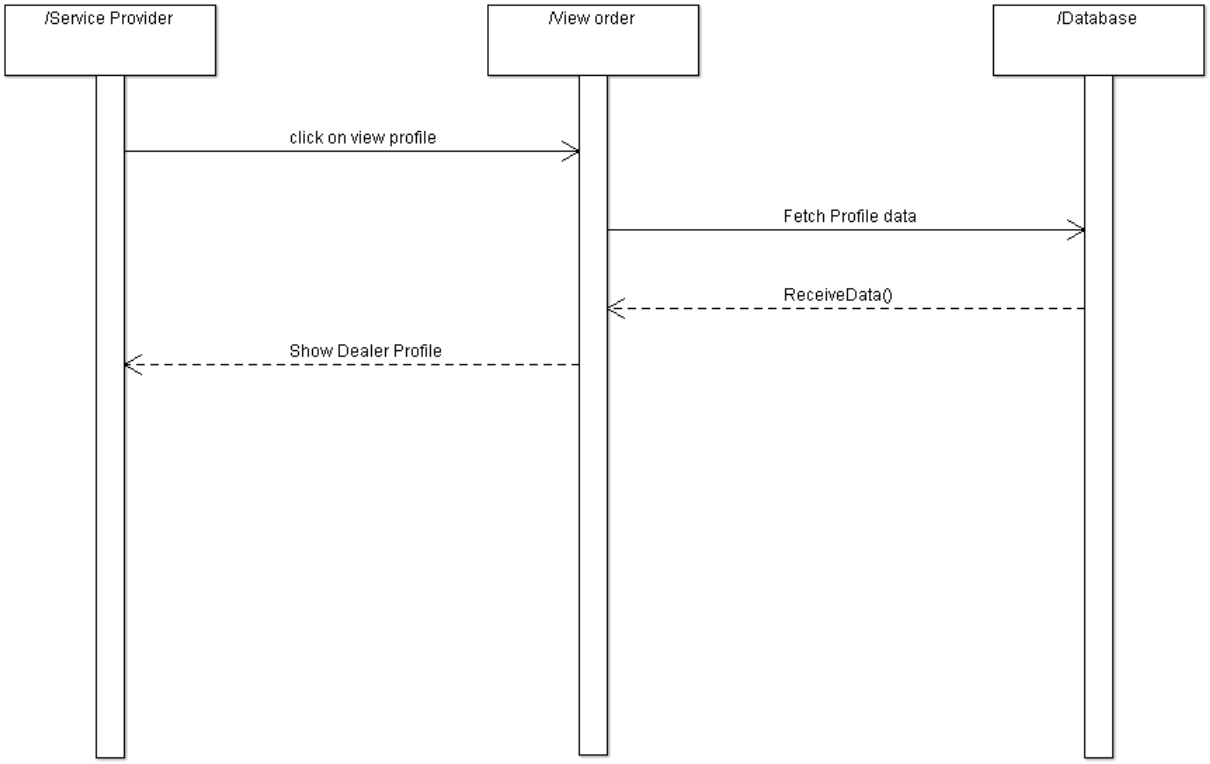
Service Provider Login


/Service_provider
/ADMIN
/Log in page
/server
loop
While(domain==false)
RequestLogin()
Show Response
Show Account List
Select Google Account
Chosse Account
Send Response
alt
if(Domain==true)
Response Successful
Open Student Login Page
alt
if(Domain == false)
Unsuccessful
Login fai
loop
While (athLOGin == false)
LogIn Data(UserID String, Password String)
Send LogIn data
alt
if(athLOGin == true)
Successful Login
Open Admin Landing Page
alt
if (athLOGin== false)
Unsuccessful Login
Login Failed

Service Provider View Analytics

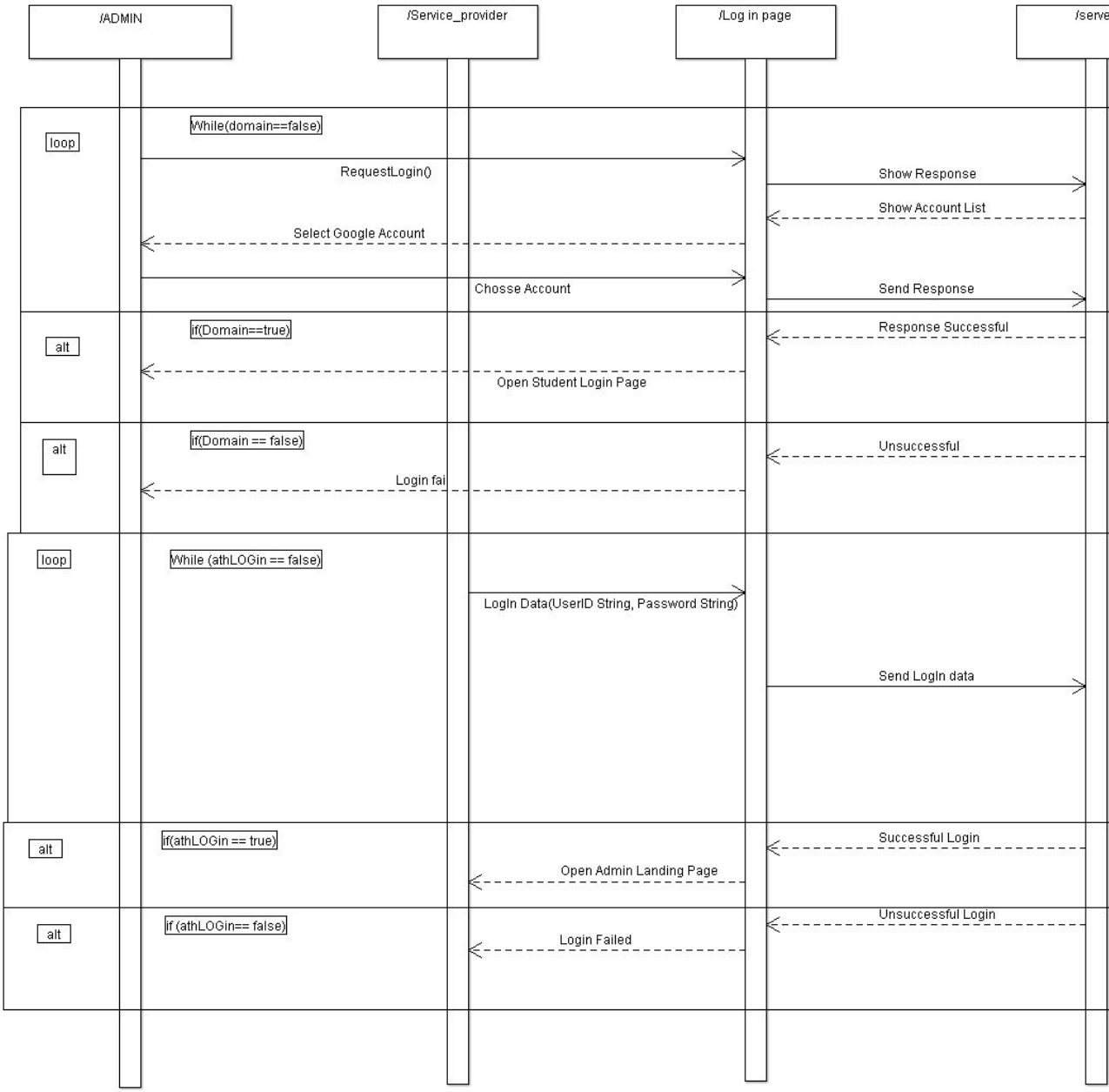


Service Provider View Order



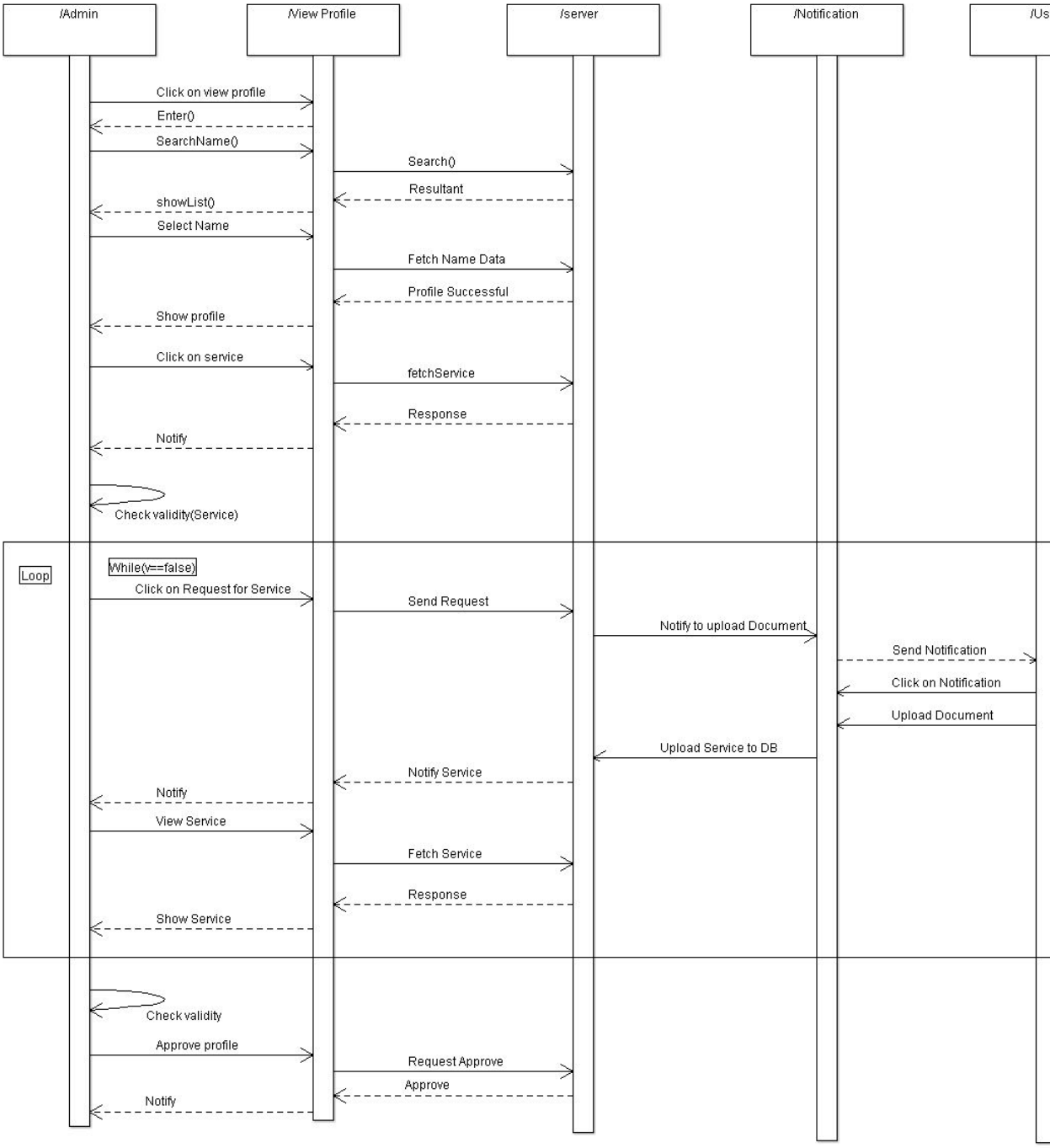
Admin:-

Admin Login

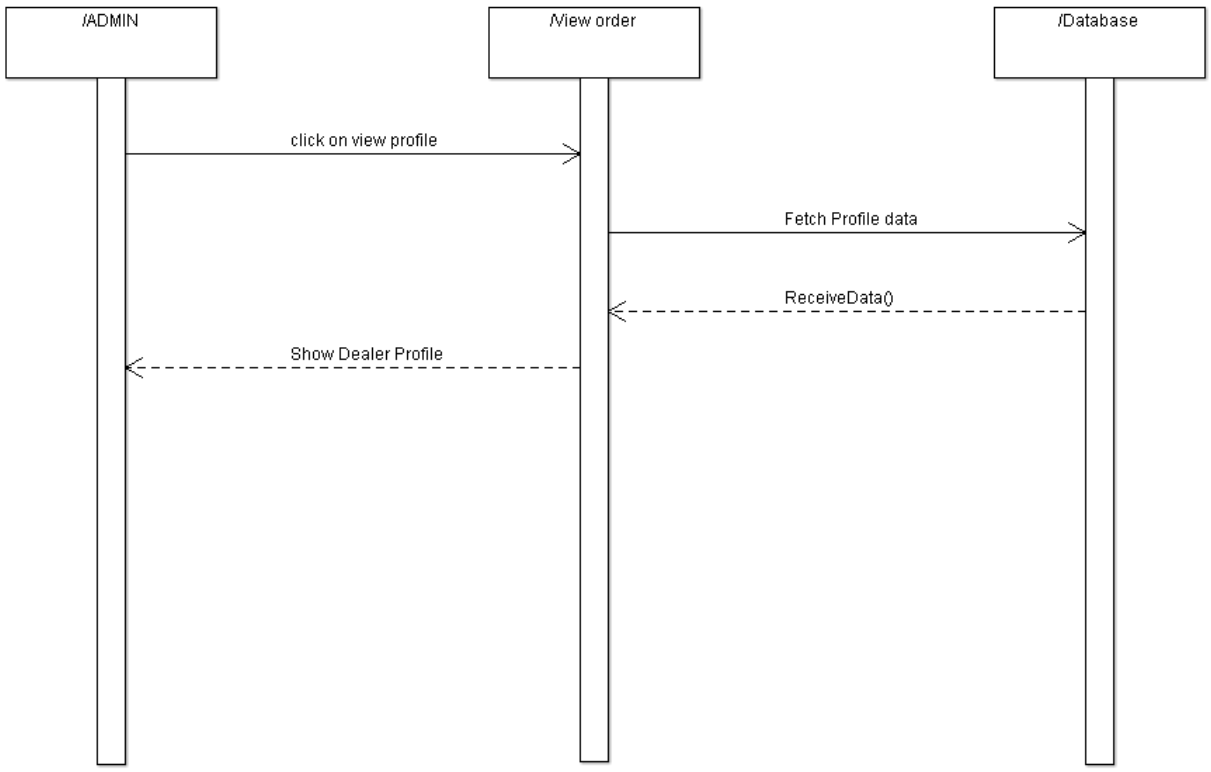


Admin Display Profile Details &

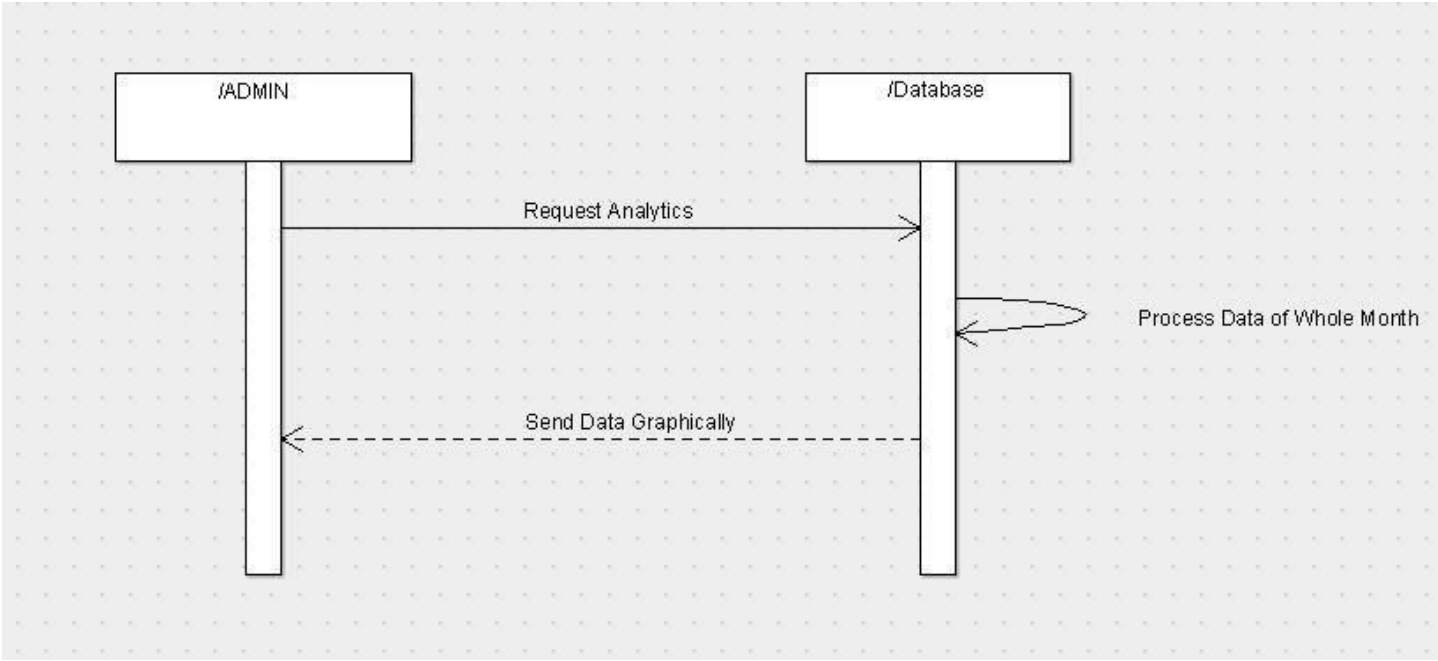
Update Details



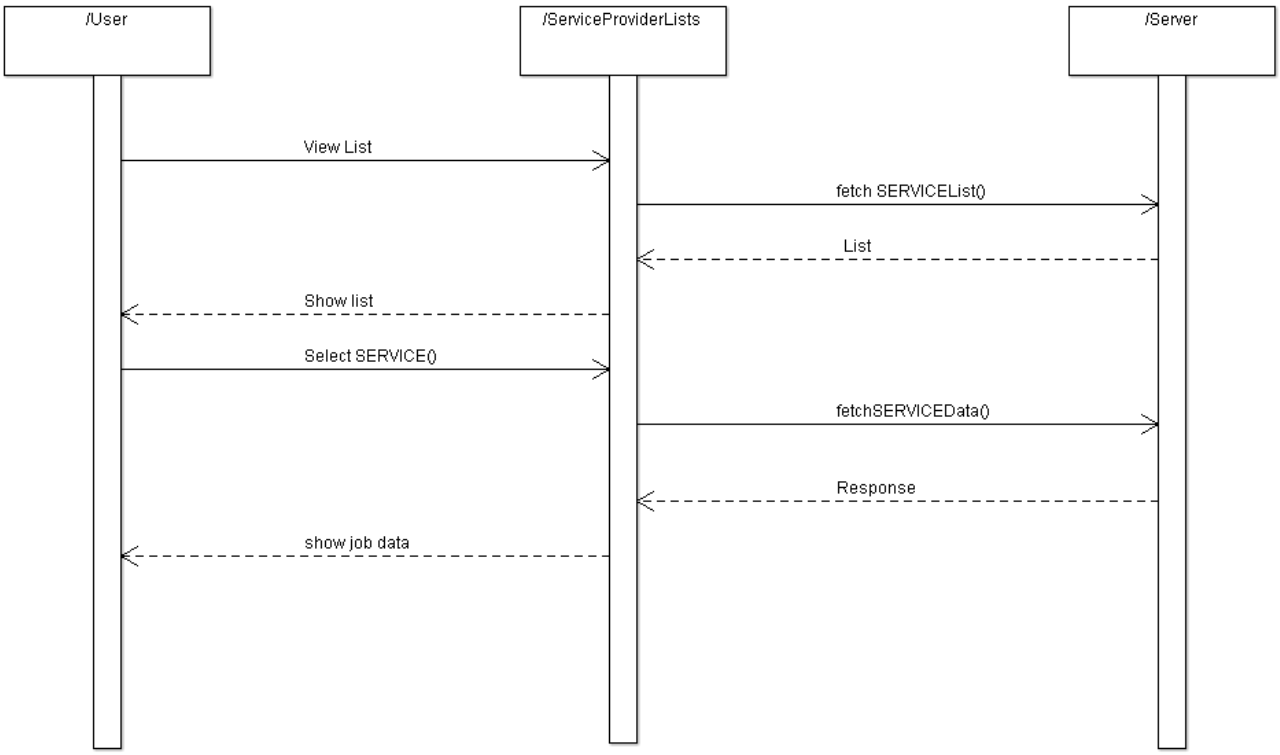
Admin View Order



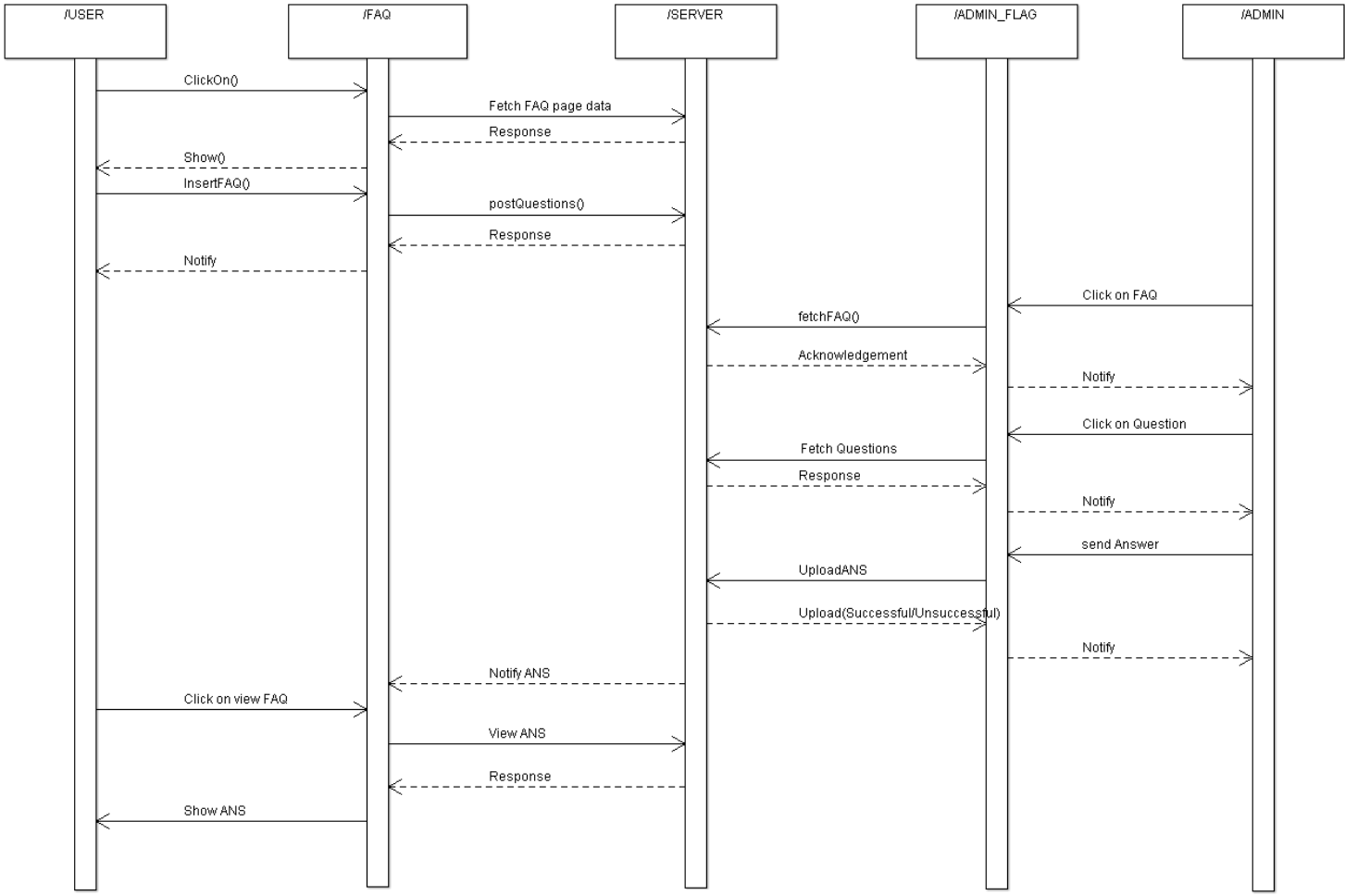
Admin View Analytics



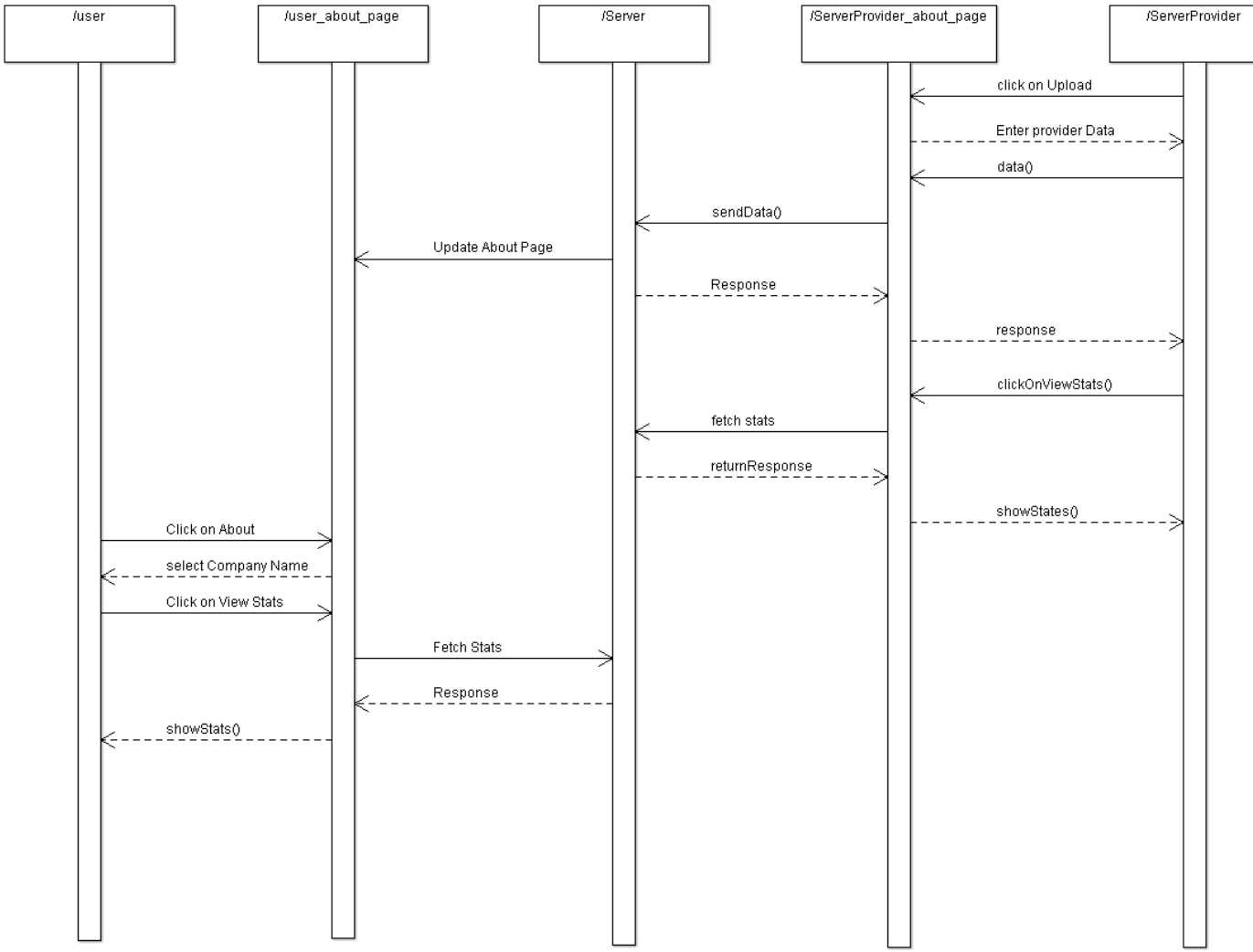
Service List:



FAQ:

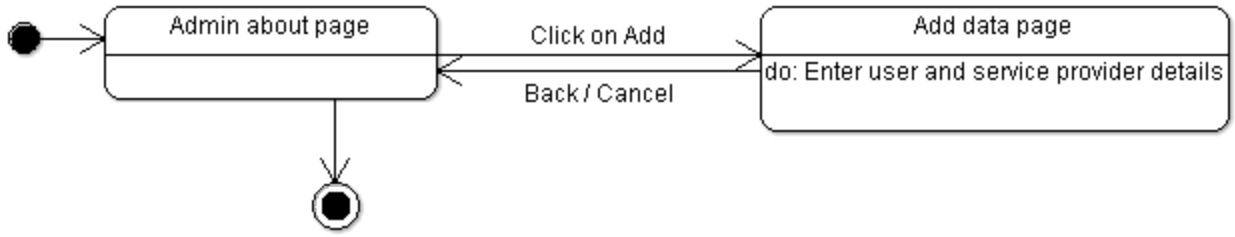

/USER
/FAQ
/SERVER
/ADMIN_FLAG
/ADMIN
ClickOn()
Fetch FAQ page data
Response
Show()
InsertFAQ()
postQuestions()
Response
Notify
Click on FAQ
fetchFAQ()
Acknowledgement
Notify
Click on Question
Fetch Questions
Response
Notify
send Answer
UploadANS
Upload(Successful/Unsuccessful)
Notify
Notify ANS
Click on view FAQ
View ANS
Response
Show ANS

About:

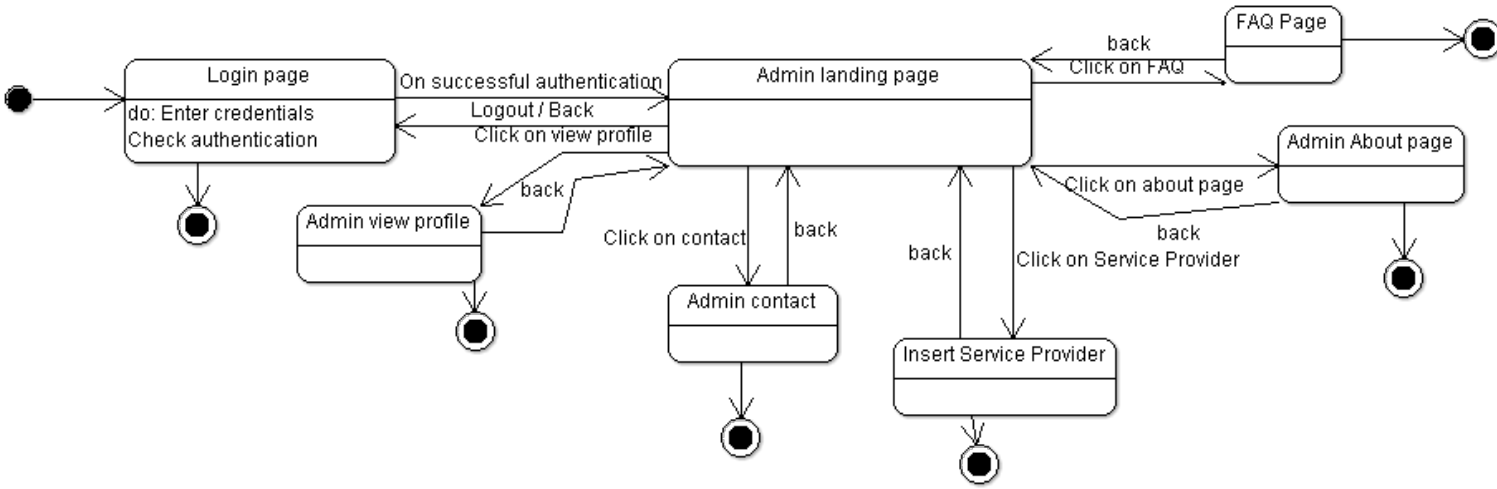

/user
/user_about_page
/Server
/ServerProvider_about_page
/ServerProvider
click on Upload
Enter provider Data
data()
sendData()
Update About Page
Response
response
clickOnViewStats()
fetch stats
returnResponse
showStates()
Click on About
select Company Name
Click on View Stats
Fetch Stats
Response
showStats()

State Diagram:

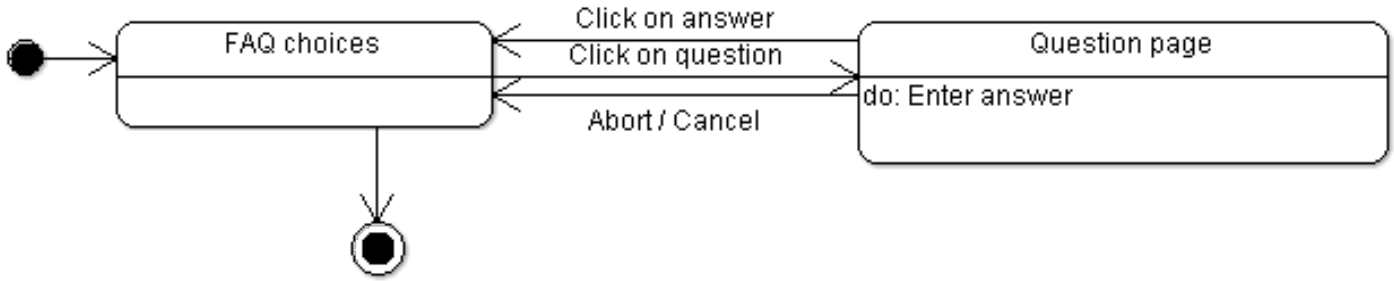
Admin About:



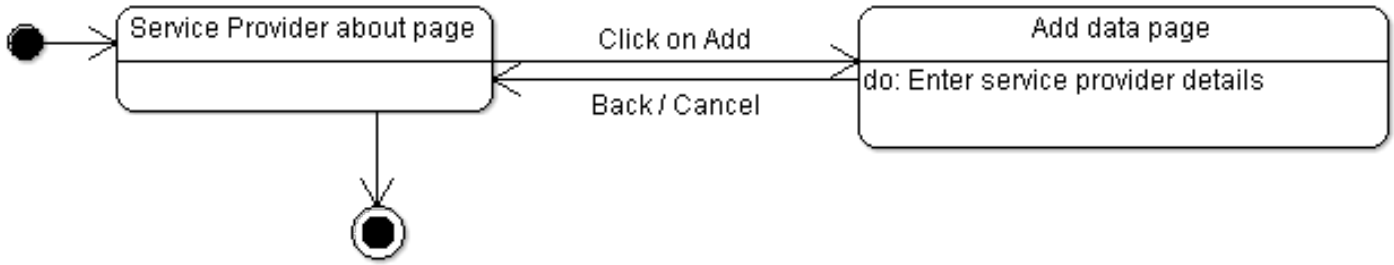
Admin Login Page:



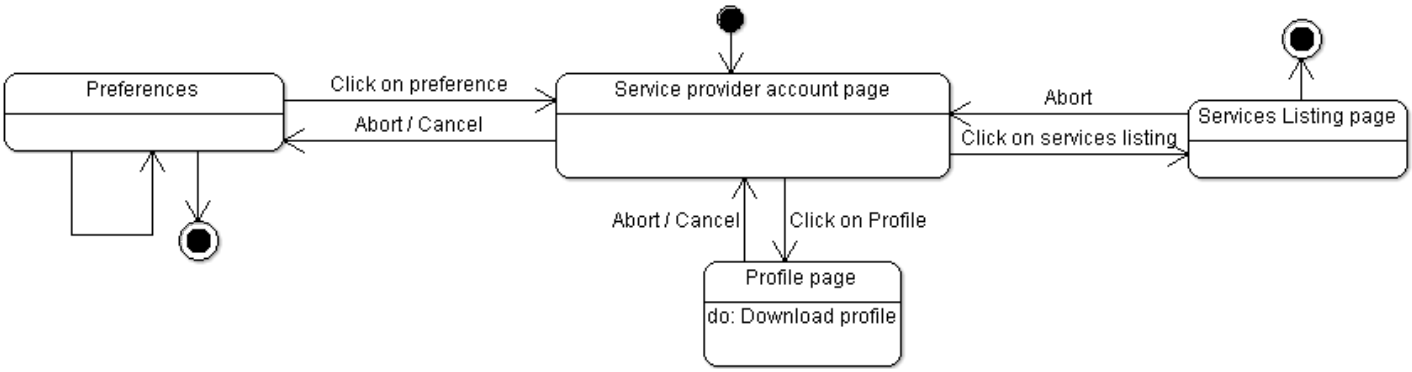
Admin FAQ:



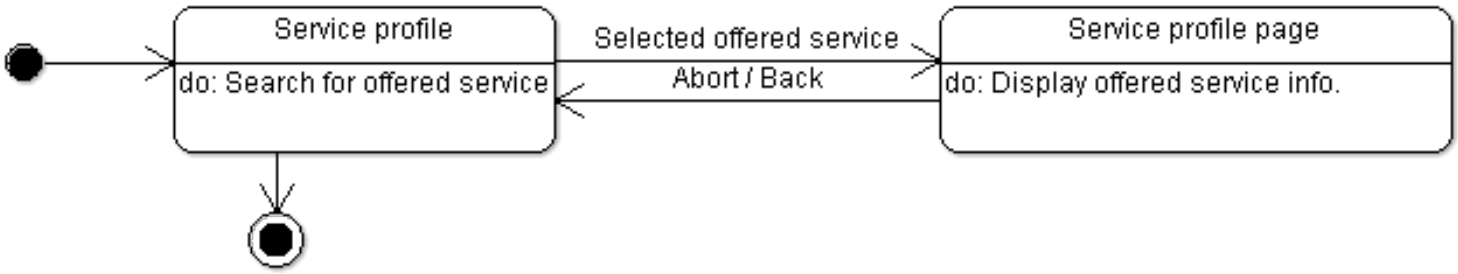
Service Provider About:



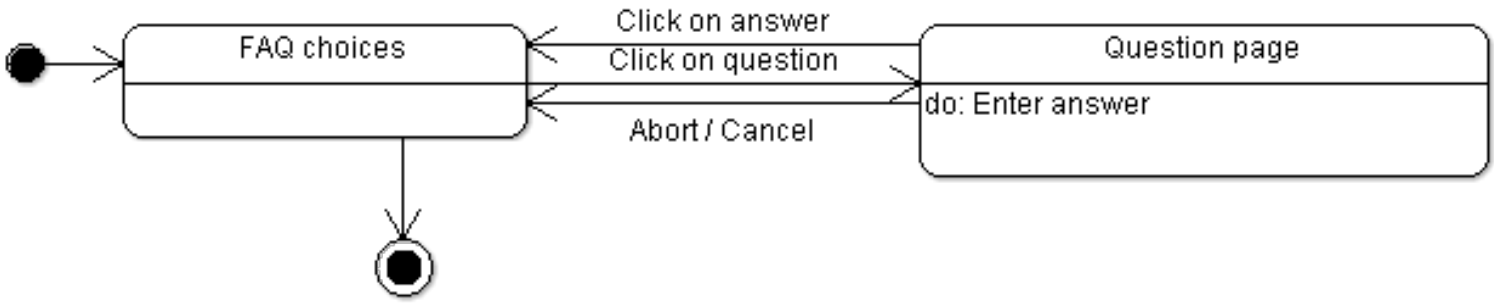
Service Provider Account Page:



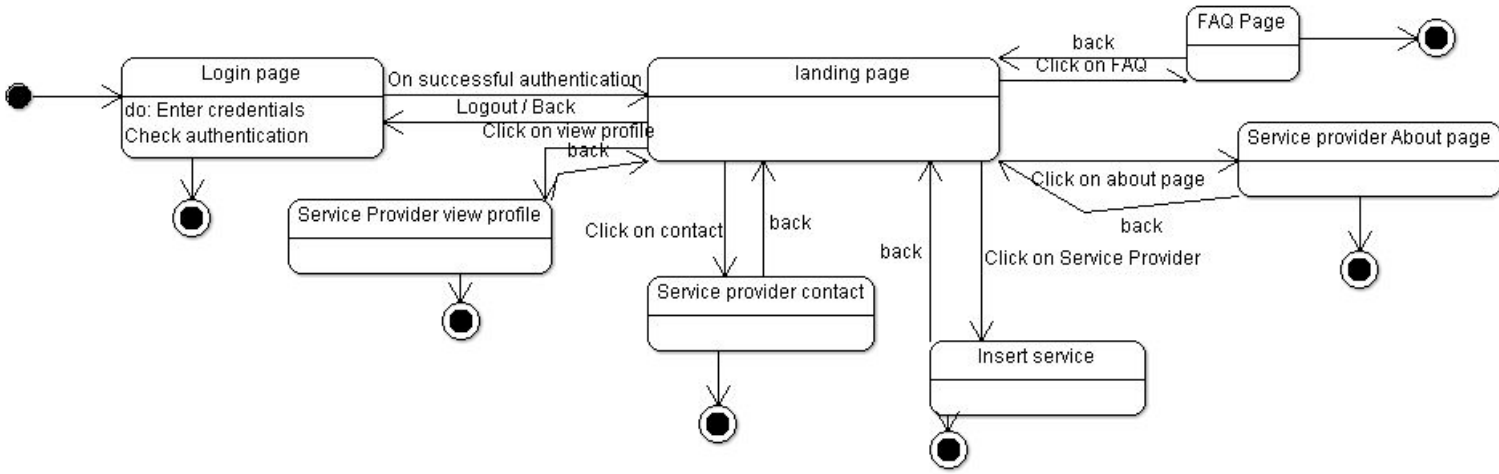
Service Provider View Service:



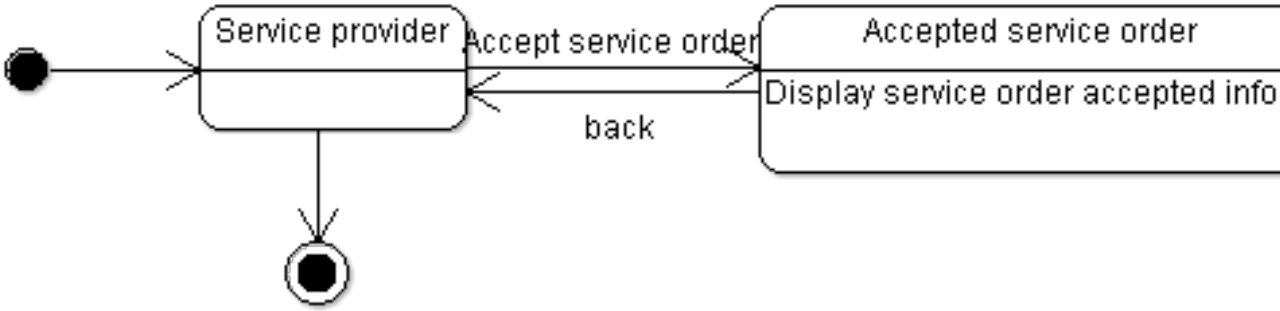
Service Provider FAQ:



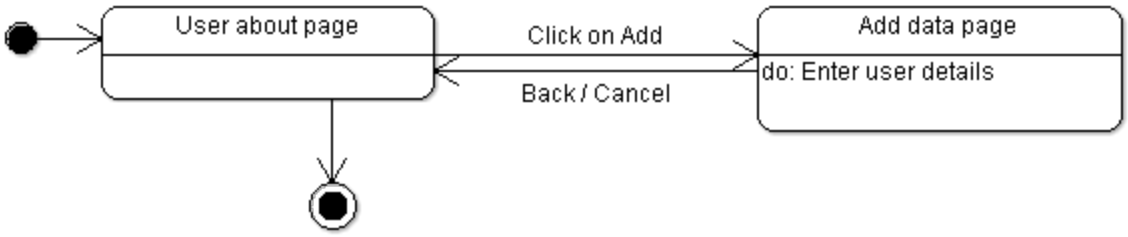
Service Provider Login Page:



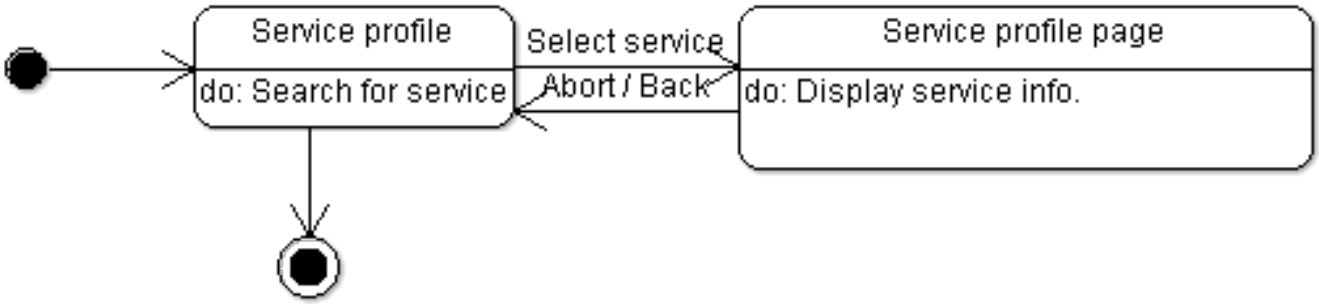
Service Provider
Service Order Accepted:



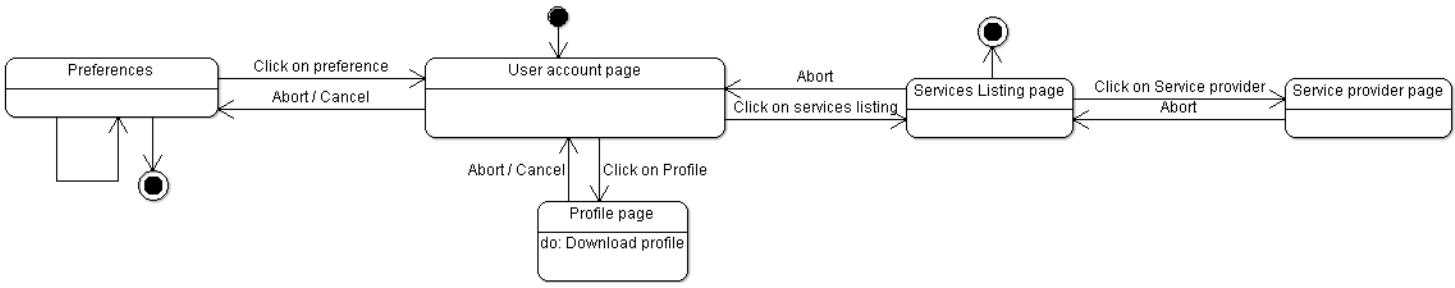
User About:



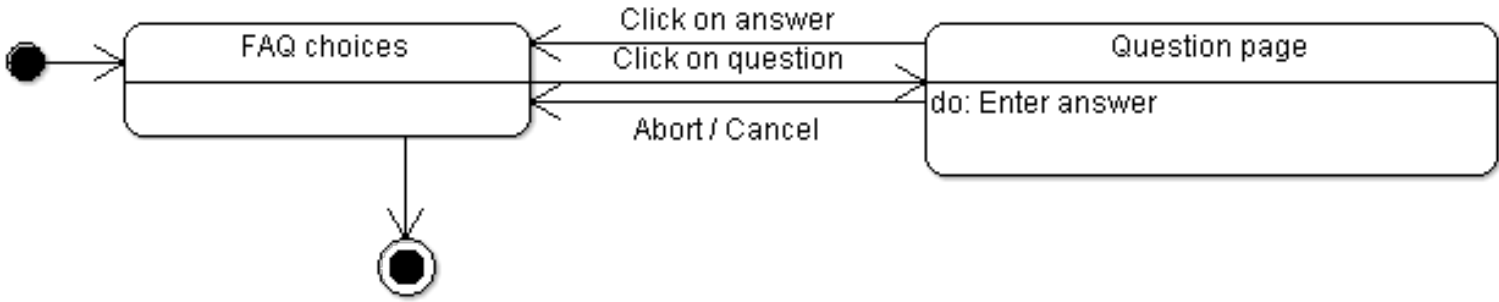
User View Service:



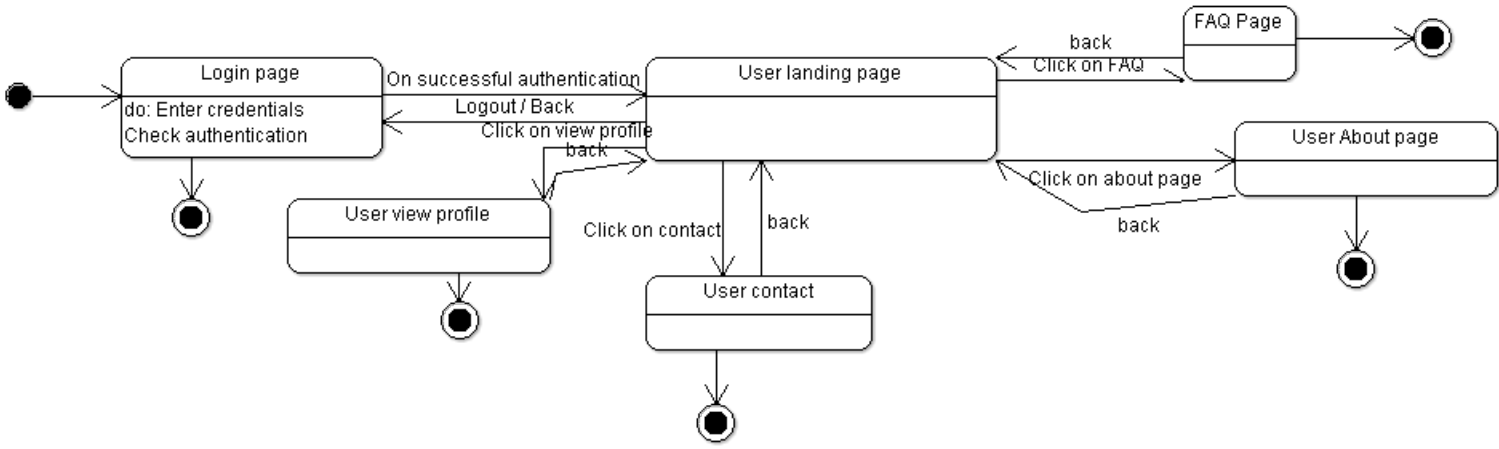
User Account Page:



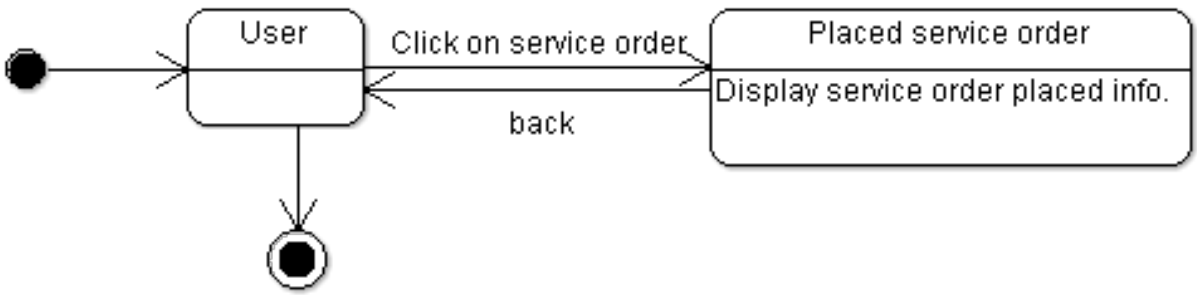
User FAQ:



User Login Page:



User Service Order:



3.1.2 Sequence Diagram:

Arrow line signifies there is a send message taken place. Response is being shown by dotted arrows.

3.1.2.1 LogIn:

It allows user to login with mail ,domain and Admin to login with the username and password that are being registered in the database already. It's loops being on same page until the correct information is not given.

3.1.2.2 Sign-up:

the user is not logged in , the user can log in as Service Provider, User or Admin.

3.1.2.3 View Profile :

It allows user to view his profile which includes his details.

3.1.2.4 Edit Profile :

It allows user to edit his profile information.

3.1.2.5: Place Orders :

It allows user to place orders.

3.1.2.6: View Orders:

It allows user to view orders placed.

3.1.2.7 : Analytics :

It allows Service Provider to analyze the sale of Service graphically.

3.1.3 State Diagram

Initial state is being shown by starting with a black dot. Final State is being shown by the black dot surrounded by an empty circle.

3.1.3.1 Service Provider LoginIn Page:

On the Login page Service Provider will enter his or her details that will include phone number and password , after entering the login details when the user will click on login button and then user will be transferred on his dashboard page.

3.1.3.2 User LoginIn Page:

On the Login page User will enter his or her details that will include phone number and password , after entering the login details when the user will click on login button and then user will be transferred on his dashboard page.

3.1.3.3 Admin LoginIn Page:

On the Login page Admin will enter his or her details that will include phone number and password , after entering the login details when the user will click on login button and then user will be transferred on his dashboard page.

3.1.3.4 SignUp :

If the user is not logged in , he or she will click on the signUp button which will transfer the user to a to page where the user is supposed to fill all the necessary details i.e name, address , phone number and password, after filling the required information to sign up , the user will click on the signup button which will register the user as Service Provider, User or Admin into the database.

3.1.3.5 Service Provider ViewProfile:

On clicking the ViewProfile button the Service Provider can view his or her details which will include all the necessary details i.e name, address and phone number.

3.1.3.6 User ViewProfile:

On clicking the ViewProfile button the User can view his or her details which will include all the necessary details i.e name, address and phone number.

3.1.3.7 Admin ViewProfile:

On clicking the ViewProfile button the Admin can view his or her details which will include all the necessary details i.e name, address and phone number.

3.1.3.8 Service Provider EditProfile:

On clicking the EditProfile button the Service Provider can edit his or her details.

3.1.3.9 User EditProfile:

On clicking the EditProfile button the User can edit his or her details.

3.1.3.10 Admin EditProfile:

On clicking the EditProfile button the Admin can edit his or her details.

3.1.3.11 PlaceOrders(Service Provider) :

On clicking the PlaceOrders button the Service Provider can view all the orders placed.

3.1.3.12 PlaceOrders(User) :

On clicking the PlaceOrders button the User can view all the orders placed.

3.1.3.13 ViewService(Service Provider):

On clicking the ViewService button the Service Provider can view Services.

3.1.3.14 ViewService(User):

On clicking the ViewService button the User can view Services.

3.1.3.15 Services:

On clicking the Service button the Service Provider can see Services.

3.1.3.19 Analytics :

On clicking the analytics button the user can view and analyse the sales of Service geographically.

3.2 Class Name: Login

Description :

The Class allows the user to enter the System by authenticating the entered credentials.

3.2.1 Method 1:checkCrediantials()

Method Description:

When the User Enters the Credentials this method will be called. Inside this method it will check weather the username and password entered is there in the database and correct.Since it is a login activity so several state may occur like

- When user is signed out

- After Login is successful

3.2.2 Method 2:Sign Up()

Method Description:
When the User clicks the ‘not a user’ link then this method is invoked and takes the user to the Sign Up Page.

3.3 Class Name:Sign Up

Description : The Class allows the user to enter the Credentials in the System and Sign Up for the Application and store the user information.

3.3.1 Method 1:register_User()

Method Description:
This method stores the user information in the firebase and is invoked when Sign Up button is clicked.

3.4 Class Name: Service Provider Page

Description : This Class Opens the Main Service Provider Page And Enables the User the to use various options like place Orders, View Orders, View Analytics, etc

3.4.1 Method 1:ViewOrders()

Input: Click View Orders Button

Output: Open View Orders Page

Method Description:
This Method Lets the Main Service Provider to View the Orders placed by the Sub Service Provider, See the Sub Service Providers name and the Details of the orders placed by the Sub Service Provider.

3.4.2 Method 2: View Service()

Input: Click View Service Button

Output: Open Service Page

Method Description:

This Method Lets the Main Service Provider to View the total Service which is left that is the total Service which is available with the Service Provider to distribute to the Service Provider.

3.4.4 Method 4:View Analytics()

Input: Click View Analytics Button

Output: Open Analytics Layout

Method Description:
This Method is invoked when the Analytics Button is Placed. This Method will Process the Data and Show the Total Sales of the Service Graphically.

3.4.5 Method 5:View Profile()

Method Description:
This Option is Available on the View Profile Page and lets the Service Provider see his Credentials entered during the time of Sign Up.

3.4.6 Method 5:Edit Profile()

Method Description:
This Option is Available on the View Profile Page and lets the Service Provider to Edit His/Her Profile accordingly and let the User Save the Edited Credentials.

3.5 Class Name: Admin Page

Description : This Class Opens the Admin Page And Enables the User the to use various options like place Orders, View Orders, View Analytics, etc

3.5.1 Method 1:ViewOrders()

Input: Click View Service Button

Output: Open View Orders Page

Method Description:
This Method lets the Admin to View the Orders placed by him and the Details of the orders.

3.5.2 Method 2:View Profile()

Method Description:
This Option is Available on the View Profile Page and lets the Admin see his Credentials entered during the time of Sign Up.

3.5.3 Method 3:Edit Profile()

Method Description:
This Option is Available on the View Profile Page and lets the Admin to Edit His/Her Profile accordingly and let the User Save the Edited Credentials.

3.5.4 Method 4:EnterLocation()

Input: Click Enter Location Button

Output: Save Admin Information

Method Description:
This Method Will Store the Location of the Admin in the Database and Be Used for the Future to View The Admin's Location.

3.5.5 Method 5: place Order() :

Input : Click place order button

Output : Open place order page

Method Description :
This method when invoked will allow the Admin to place an order.

3.6 Class Name: Service Provider Edit Profile

Description : This class opens the Edit profile page and enables the Service Provider to edit his profile information which includes his name, address, password and etc.

3.6.1 Method 1: editName() Input:

Click Edit Profile Button.

Output : Open Edit Profile Page.

Method Description:

This method lets the Service Provider to edits his name.

3.6.2 Method 2: editPhoneNo()

Input: Click Edit Profile Button.

Output : Open Edit Profile Page.

Method Description:

This method lets the Service Provider to edits his phone number.

3.6.3 Method 2: editGender()

Input: Click Edit Profile Button.

Output : Open Edit Profile Page.

Method Description:

This method lets the Service Provider to edits his/her gender.

3.6.4 Method 2: editAddress()

Input: Click Edit Profile Button.

Output : Open Edit Profile Page.

Method Description:

This method lets the Service Provider to updates his address.

3.6.5 Method 2: **editPassword()**

Input: Click Edit Profile Button.

Output : Open Edit Profile Page.

Method Description:
This method lets the Service Provider to edit his password.

3.7 Class Name:User Edit Profile

Description : This class opens the Edit profile page and enables the User to edit his profile information which includes his name,address,password and etc.

3.7.1 Method 1: editName() Input:
Click Edit Profile Button.

Output : Open Edit Profile Page.

Method Description:
This method lets the User to edits his name.

3.7.2 Method 2: **editPhoneNo()**

Input: Click Edit Profile Button.

Output : Open Edit Profile Page.

Method Description:
This method lets the User to edits his phone number.

3.7.3 Method 2: **editGender()**

Input: Click Edit Profile Button.

Output : Open Edit Profile Page.

Method Description:
This method lets the User to edits his/her gender.

3.7.4 Method 2: **editAddress()**

Input: Click Edit Profile Button.

Output : Open Edit Profile Page.

Method Description:
This method lets the User to updates his address.

3.7.5 Method 2: **editPassword()**

Input: Click Edit Profile Button.

Output : Open Edit Profile Page.

Method Description:
This method lets the User to edit his password.

3.8 Class Name:Admin Edit Profile

Description : This class opens the Edit profile page and enables the Admin to edit his profile information which includes his name,address,password and etc.

3.8.1 Method 1: editName() Input:
Click Edit Profile Button.

Output : Open Edit Profile Page.

Method Description:
This method lets the Admin to edits his name.

3.8.2 Method 2: **editPhoneNo()**

Input: Click Edit Profile Button.

Output : Open Edit Profile Page.

Method Description:
This method lets the Admin to edits his phone number.

3.8.3 Method 2: **editGender()**

Input: Click Edit Profile Button.

Output : Open Edit Profile Page.

Method Description:
This method lets the Admin to edits his/her gender.

3.8.4 Method 2: **editAddress()**

Input: Click Edit Profile Button.

Output : Open Edit Profile Page.

Method Description:
This method lets the Admin to updates his address.

3.8.5 Method 2: **editPassword()**

Input: Click Edit Profile Button.

Output : Open Edit Profile Page.

Method Description:
This method lets the Admin to edit his password.

3.9 Class Name: Sub Service Provider Page

Description : This Class Opens the Sub Service Provider Page And Enables the User the to use various options like place Orders, View Orders of the Admin, View Profile,Edit Profile , etc

3.9.1 Method 1:ViewOrders()

Input: Click View Orders Button

Output: Open View Orders Page

Method Description:

This Method Lets the Sub Service Provider to View the Orders placed by the Admin, See the Admin's name and the Details of the orders placed by the Admin.

3.9.2 Method 2:View Service()

Input: Click View Service Button

Output: Open Service Page

Method Description:

This Method Lets the Sub Service Provider to View the total Service which is left that is the total Service which is available with the SubService Provider and when to place an order to the Service Provider.

3.9.3 Method 5:Edit Profile()

Method Description:

This Option is Available on the View Profile Page and lets the Sub Service Provider to Edit His/Her Profile accordingly and let the User Save the Edited Credentials.

3.9.4 Method 5:View Profile()

Method Description:

This Option is Available on the View Profile Page and lets the Sub Service Provider see his Credentials entered during the time of Sign Up.

3.10 Class Name:Service Provider View Profile

Description :

This class opens the View profile page and enables the Service Provider to view his profile information which includes his name,address,password and etc.

3.10.1 Method 1: viewName()

Input: Click View Profile Button.

Output : Open View Profile Page.

Method Description:

This method lets the Service Provider to view his name.

3.10.2 Method 2 :

viewPhoneNo() Input: Click View Profile Button.

Output : Open View Profile Page.

Method Description:

This method lets the Service Provider to view his phone number.

3.10.3 Method 2: **editGender()**

Input: Click View Profile Button.

Output : Open View Profile Page.

Method Description:

This method lets the Service Provider to view his/her gender.

3.10.4 Method 2: **ViewAddress()**

Input: Click View Profile Button.

Output : Open View Profile Page.

Method Description:

This method lets the Service Provider to view his address.

3.10.5 Method 2: **viewPassword()**

Input: Click view Profile

Button.

Output : Open View Profile Page.

Method Description:
This method lets the Service Provider to view his password.

3.11 Class Name:User View Profile

Description : This class opens the View profile page and enables the SubService Provider to view his profile information which includes his name,address,password and etc.

3.10.1 Method 1: viewName()
Input: Click View Profile Button.

Output : Open View Profile Page.

Method Description:
This method lets the SubService Provider to view his name.

3.10.2 Method 2: **viewPhoneNo()**

Input: Click View Profile Button.

Output : Open View Profile Page.

Method Description:
This method lets the SubService Provider to view his phone number.

3.10.3 Method 2: **editGender()**

Input: Click View Profile Button.

Output : Open View Profile Page.

Method Description:
This method lets the SubService Provider to view his/her gender.

3.10.4 Method 2: **ViewAddress()**

Input: Click View Profile Button.

Output : Open View Profile Page.

Method Description:
This method lets the SubService Provider to view his address.

3.10.5 Method 2:**viewPassword()**

Input: Click view Profile Button.

Output : Open View Profile Page.

Method Description:
This method lets the SubService Provider to view his password.

3.12 Class Name:Admin View Profile

Description :
This class opens the View profile page and enables the Admin to view his profile information which includes his name,address,password and etc.

3.10.1 Method 1: viewName()
Input: Click View Profile Button.

Output : Open View Profile Page.

Method Description:
This method lets the Admin to view his name.

3.10.2 Method 2: **viewPhoneNo()**

Input: Click View Profile Button.

Output : Open View Profile Page.

Method Description:
This method lets the Admin to view his phone number.

3.10.3 Method 2: **editGender()**

Input: Click View Profile Button.

Output : Open View Profile Page.

Method Description:
This method lets the Admin to view his/her gender.

3.10.4 Method 2: **ViewAddress()**

Input: Click View Profile Button.

Output : Open View Profile Page.

Method Description:
This method lets the Admin to view his address.

3.10.5 Method 2:**viewPassword()**

Input: Click view Profile Button.

Output : Open View Profile Page.

Method Description:
This method lets the Admin to view his password.

3.13 Class Name:View Orders

Description:

This Class will allow the Service Provider, Sub Service Provider and The Admin to view his/her Orders and the Quantity of Service Places .

3.13.1 Method 1:show_order_list()

Input: User Clicks View Order Button

Output: Open A List of Orders

Method Description

This Method when invoked will allow the Service Provider, Sub Service Provider and The Admin to view his/her Orders and the Quantity in the Form of List and will show the details of the item /Order Placed

3.14 Class Name: Place Orders

Description:

This class opens the Place order page and enables the User to place an order to Service Provider and Admin to place an order to User.

3.14.1 Method 1: placeOrders() :

Input : Click place order button

Output : Open place order page

Method Description :

This method when invoked will allow the User and Service Provider to place an order.

3.18 Class Name : View Services()

Description:

This Class will Help to View the total Service Available with the Service Provider and subService Provider.

3.18.1 Method Name:totalServiceAvailable()

Input: click totalServiceAvailable page

Output: Open total Service available page

Method Description:
This method will view the total Service available with the Service Provider and SubService Provider

3.19 Class Name : View Analytics()

Description :
This class enables the Service Provider to show the sales of Service graphically.

3.19.1 Method 1: analyticDataInfo()

Input : Click analyticDataInfo page.

Output : Open graph Analytic page

Method Description:

This method will analyze the sale of Service and according to the data it will shows the sale of Service in different regions graphically.

5.0 Design decisions and tradeoffs

The design decision to use three screens separately for Service Provider and User for better user interface. It will help in providing privacy to the Service Provider and the User.
A possible tradeoff when considering links is to use buttons instead of items in the menu. This design decision - to use buttons for navigating between screens - is to enhance visibility. Therefore, it is easier for the user to Navigate Through the Pages.

6.0 Pseudocode for Components

```
#include <iostream>


using namespace std;


class ViewAdminProfile{

string str, br,inSring;

void onCreate();

void onItemClick();
```

```
void onClick();
}

class ServiceData: public ViewAdminProfile{
int url,name;
void getServiceUrl();
void getServiceName();
void setServiceUrl(sring name);
void setServiceName(string name);
}

class ServiceDataAdapter: public ViewAdminProfile{
int layoutresource;
void getView(int position);
}

class AdminAbout{
string info,url;
void onCreate();
void updateCompanyDetails();
}

class ServiceListings{
string result;
void onCreateDialog();
}

class UserAccount{
void onClick();
void onCreateDialog();
```

```
}

class UserAbout{
string servName;
void onCreate();
}

class FAQAdmin{
string jsonurl;
void showQuestion();
void onClick();
void answerQuestions(int qID);
}

void UserLanding{
string mActivityTitle;
void onCreate();
void onStart();
}

class LogIn{
int rcSignin;
void onCreate();
void signIn();
void onCompletetext();
void onStart();
void view();
void userLogin();
void view();
}
```

```
class ViewProfileUser{
string var;
string inpString;
void onClick();
void onCreateDialog();
}

class AboutService{
string compRegistration;
string getCompData();
void showInList();
void onClick();
void checkApproval();
}

class Home{
string servProvidername;
string url;
string getCompaniesNames();
void applyForCompanies();
void onClick();
}

class UserFAQ{
string variable;
string question;
void askQuestion();
void onCreateDialog();
}
```

```
class Notification{
string title;
string urlTest;
void onCreateDialog();
void onClick();
void notification();
void viewNotification();
}

class AppCompatActivity: public AdminAbout,public UserAbout,public FAQ_Admin,
public UserLanding, public Login,public Notification,public UserFAQ, public Home,
public ViewProfile_User,public ViewAdminProfile,public ServiceListings {
int newAttr;
void onStart();
void onCreate();
void onActivityResult();
void newOperation();
void onPastCreate();
void onOptionsItemsSelected();
}

int main() {

  return 0;
}
```

7.0 Coding Metrics

CCCC Software Metrics Report Summary table of high level measures summed over all files processed in the current run. Table of procedural measures (i.e. lines of code, lines of comment, McCabe's cyclomatic complexity summed over each module. Table of four of the 6 metrics proposed by Chidamber and Kemerer in their various papers on 'a metrics suite for object oriented design'. Structural metrics based on the relationships of each module with others. Includes fan-out (i.e. number of other modules the current module uses), fan-in (number of other modules which use the current module), and the Information Flow measure suggested by Henry and Kafura, which combines these to give a measure of coupling for the module. Lexical counts for parts of submitted source files which the analyser was unable to assign to a module. Each record in this table relates to either a part of the code which triggered a parse failure, or to the residual lexical counts relating to parts of a file not associated with a specific module. A description of the CCCC program.
generated Mon Nov 12 23:48:55 2018

Project Summary

This table shows measures over the project as a whole.

- NOM = Number of modules
  Number of non-trivial modules identified by the analyser. Non-trivial modules include all classes, and any other module for which member functions are identified.
- LOC = Lines of Code
  Number of non-blank, non-comment lines of source code counted by the analyser.
- COM = Lines of Comments
  Number of lines of comment identified by the analyser
- MVG = McCabe's Cyclomatic Complexity
  A measure of the decision complexity of the functions which make up the program.The strict definition of this measure is that it is the number of linearly independent routes through a directed acyclic graph which maps the flow of control of a subprogram. The analyser counts this by recording the number of distinct decision outcomes contained within each function, which yields a good approximation to the formally defined version of the measure.
- L_C = Lines of code per line of comment
  Indicates density of comments with respect to textual size of program
- M_C = Cyclomatic Complexity per line of comment
  Indicates density of comments with respect to logical complexity of program
- IF4 = Information Flow measure
  Measure of information flow between modules suggested by Henry and Kafura. The analyser makes an approximate count of this by counting inter-module couplings identified in the module interfaces.

Two variants on the information flow measure IF4 are also presented, one (IF4v) calculated using only relationships in the visible part of the module interface, and the other (IF4c) calculated using only those relationships which imply that changes to the client must be recompiled of the supplier's definition changes.

Metric	Tag	Overall	Per Module
Number of modules	NOM	1	
Lines of Code	LOC	6	6.000
McCabe's Cyclomatic Number	MVG	1	1.000
Lines of Comment	COM	0	0.000
LOC/COM	L_C	------	
MVG/COM	M_C	------	
Information Flow measure (inclusive)	IF4	0	0.000
Information Flow measure (visible)	IF4v	0	0.000
Information Flow measure (concrete)	IF4c	0	0.000
Lines of Code rejected by parser	REJ	1	

Procedural Metrics Summary

For descriptions of each of these metrics see the information preceding the project summary table. The label cell for each row in this table provides a link to the functions table in the detailed report for the module in question

Module Name	LOC	MVG	COM	L_C	M_C
anonymous	5	1	0	------	------

Object Oriented Design

- WMC = Weighted methods per class
  The sum of a weighting function over the functions of the module. Two different weighting functions are applied: WMC1 uses the nominal weight of 1 for each function, and hence measures the number of functions, WMCv uses a weighting function which is 1 for functions accessible to other modules, 0 for private functions.
- DIT = Depth of inheritance tree
  The length of the longest path of inheritance ending at the current module. The deeper the inheritance tree for a module, the harder it may be to predict its behaviour. On the other hand, increasing depth gives the potential of greater reuse by the current module of behaviour defined for ancestor classes.
- NOC = Number of children
  The number of modules which inherit directly from the current module. Moderate values of this measure indicate scope for reuse, however high values may indicate an inappropriate abstraction in the design.
- CBO = Coupling between objects
  The number of other modules which are coupled to the current module either as a client or a supplier. Excessive coupling indicates weakness of module encapsulation and may inhibit reuse.

The label cell for each row in this table provides a link to the module summary table in the detailed report for the module in question

Module Name	WMC1	WMCv	DIT	NOC	CBO
anonymous	1	0	0	0	0

Structural Metrics Summary

- FI = Fan-in
  The number of other modules which pass information into the current module.
- FO = Fan-out
  The number of other modules into which the current module passes information
- IF4 = Information Flow measure
  A composite measure of structural complexity, calculated as the square of the product of the fan-in and fan-out of a single module. Proposed by Henry and Kafura.

Note that the fan-in and fan-out are calculated by examining the interface of each module. As noted above, three variants of each each of these measures are presented: a count restricted to the part of the interface which is externally visible, a count which only includes relationships which imply the client module needs to be recompiled if the supplier's implementation changes, and an inclusive count The label cell for each row in this table provides a link to the relationships table in the detailed report for the module in question

Module Name	Fan-out			Fan-in			IF4			vis	con	inc	vis	con	incl	vis	con	inc
anonymous	0	0	0	0	0	0	0	0	0	0								

Other Extents

Location	Text	LOC	COM	MVG
main.cpp:1	<file scope items>	1	0	0

About CCCC

This report was generated by the program CCCC, which is FREELY REDISTRIBUTABLE but carries NO WARRANTY.

CCCC was developed by Tim Littlefair, as part of a PhD research project. This project is now completed and descriptions of the findings can be accessed at http://www.chs.ecu.edu.au/~tlittlef

User support for CCCC can be obtained by mailing the list cccc-users@lists.sourceforge.net.

Please also visit the CCCC development website at http://cccc.sourceforge.net.

8.0 Minutes of the Meeting with Service Providers

Minutes of Meeting

Date: 10/11/2018

Time: 10:30 AM

Telephonic Meeting.

Meeting called by:	Q-Taskers
Meeting for:	Service Provider
Type of Meeting:	Telephonic
Name of the shop:	Jai Durge Electronics
Address:	E-312, Barra-4, Kanpur, 208027
Shopkeeper Name:	Akhilesh Kumar

Attendees:

1. Karan Inder Singh (U101116FCS057)
2. Rahul Saha (U101116FCS241)
3. Rajat Srivastava (U101116FCS098)
4. Rishabh Gaur (U101116FCS100)
5. Akhlesh Kumar

Minutes

Agenda items:

1. To register 'Jai Durge Electronics' to the website and fill the 'join us' form for them.
2. To discuss with them about the business model and the revenue model of the Q-Taskers.
3. To ask for their valuable opinions or feedback.

Conclusions:

Jai Durge Electronics (shop) has been registered as "service provider" for the Q-Taskers, for a tenure of next 6 months.

Action items:

- ✓ Jai Durge Electronics will work for Q-Taskers as "service provider" for a period of next 6 months.
- ✓ Jai Durge Electronics will repair "lights" and deal with "house wiring system".
- ✓ Jai Durge Electronics will work within the 5 Km area of their shop.
- ✓ Jai Durge Electronics will pay Q-Taskers Rs. 2000 as fee for 6 months.

- ✓ Every service provided by the Jai Durge Electronics will be verified through a star rating system of the Q-Taskers.
- ✓ Continuous less than 3-star rating service of Jai Durge Electronics can lead to their rejection of application as “service provider”.
- ✓ Cost of tools and transportation will be managed by the side of Jai Durge Electronics.
- ✓ Jai Durge Electronics cannot back-off from providing the services for at-least initial 6 months.
- ✓ Jai Durge Electronics will have to reach every household within 60 minutes or less or they cannot charge the visiting fee to the customers.
- ✓ Visiting fee by Jai Durge Electronics cannot exceed Rs. 150.
- ✓
- ✓
- ✓
- ✓