

INFOSTRETCH MOBIMINTED

By

Mitul A. Shah (ID No. 072140)

Harshil A. Patel (ID No. 072058)

A project submitted

In

**partial fulfillment of the requirements
for the degree of**

BACHELOR OF ENGINEERING

In

COMPUTER ENGINEERING

Internal Guide

Prof. Malay S. Bhatt

Asst. Professor

Dept. of Comp. Engg

External Guide

Mr. Ashwin Vairu

Tech. Lead

Infostretch Solutions Pvt. Ltd.



**Faculty of Technology
Department of Computer Engineering
Dharmsinh Desai University**

April 2011

Certificate

This is to certify that the project work titled

INFOSTRETCH MOBIMINTED

is the bona fide work of

Harshil A. Patel (ID No. 072058)

Mitul A. Shah (ID No. 072140)

carried out in the partial fulfillment of the degree of Bachelor of Engineering
in Computer Engineering at Dharmsinh Desai University in the academic
session December 2010 to April 2011

(Prof. Malay S. Bhatt)

Assistant Professor,

Dept. of Computer Engg.

(Prof. C.K.Bhensdadia)

Head,

Dept. of Computer Engg.



**Faculty of Technology
Department of Computer Engineering
Dharmsinh Desai University**

Infostretch MobiMinted is an e-commerce application working on Apple iPad. It concerns with designing cards for special events of life like wedding, party, announcements etc. It includes utilizing various features of iPad as well as implementation of PayPal services for online payment. User can personalize card(s) by writing on card(s).

The whole implementation of Infostretch MobiMinted is developed in Objective-C. Web-services are also developed at server side. Infostretch MobiMinted uses web-services for various requests made by user. User has full fledged to choose the type of paper he/she wants the design to be printed on. User can select quantity of cards.

The project contains many modules such as Personalize, View Template, Save Design, View saved design, Add sample etc. Personalize module provides facilities such as editing the templates, selecting quantity, selecting paper type of the card, etc. View template module provides facility to select from various templates. Save design module provides facility to save the design so that user can resume it afterwards. View saved design module provides facility to view the saved designs so that user can edit it.

This project will give end user the ease of designing cards utilizing various features of iPad.

ACKNOWLEDGEMENT

As the final frontier towards achieving a Degree, the activity of going through industrial orientation had bridged the gap between the academics and practical real-life work for me. It had prepared me to apply myself better to become good IT professional. It required a lot of people's support to complete this training.

We would like to express our heartfelt gratitude towards Infostretch Solutions Pvt. Ltd., Ahmedabad who has given us an opportunity to develop a project in the organization.

We are sincerely grateful to **Mr. Ashwin Vairu** (Technology Leader) for his consistence guidance and suggestions about technical and managerial knowledge about the project for the success of the project.

We are also grateful to **Mr. Niraj Joshi** (Team Leader) who gave us ideas and necessary technical support for completion of the project. Other team members of Infostretch Solutions Pvt. Ltd. were very helpful in developing this project. We extend our thanks to Mr. Mishal Shah, Mr. Bipin Gohel, and Mr. Javal Nanda for the same.

We are extremely grateful to our internal guide **Mr. Malay S. Bhatt** for taking active interest in progress of the project. He was always ready to help me in the entire course of my project work, in spite of his busy schedule at college.

With Sincere Regards,

Mitul Shah
Harshil Patel

1.1 ABOUT INFOSTRETCH

InfoStretch is a leader in services and solutions for Mobile Application Development and Testing, Quality Assurance Testing and Automation, SaaS solutions and ERP Testing solutions.

InfoStretch delivers high-quality, reliable and cost effective services to clients. As a dynamic, young, and innovative organization, we think outside the box and go out of our way to tackle information technology challenges. We specialize in delivering outsourced business solutions that command deep technology knowledge, process proficiency and domain expertise. We help our customers protect their brands by leveraging technology and higher quality work through our unbiased solutions using objective quality metrics as an Independent Validation & Verification (IV&V) company, achieving unprecedented levels of quality for outsourced projects. Our integrated network of offshore facilities in India is supported by onsite and offsite capabilities in the U.S. We implement aggressive SLA-based outsourcing models and strive to compress delivery timeframes for our customers.

InfoStretch was formed in 2004 by a team of highly qualified and experienced professionals known in the software QA world. The team came together with the goal to leverage their international industry experience and build a global, comprehensive services organization with the highest quality of services.

1.2 PROJECT DETAILS

Infostrech MobiMinted is a kind of e-commerce application which lets you view – personalize – order the greeting cards according to your occasion your way. To use this application u need an iPad and wifi network as the application directly communicates with server. The entire database is maintained at server side and according to query it responds with data required. GlassFish is used as server and MySQL as backend database. The total communication between client and server is made through Web service and SOAP request/reply only. This application includes features like saving design and user can also order a sample card to have idea of paper type and printing quality. The entire application is

developed in language Objective C which is widely used for developing application for iPad, iPhone and Mac OS. The server is designed in Java.

1.3 PURPOSE

Infostretch MobiMinted lets user see the different kinds of card according to different occasion. User can view, edit, save and also see previously saved designs. In a way this application lets you design whole card from choosing paper type and occasion till payment options.

1.4 SCOPE

Infostretch MobiMinted can be used by any person having iPad with wifi network available.

1.5 TOOLS AND TECHNOLOGIES

Tools : Xcode, iPad Simulator, Interface Builder, NetBeans.

Technologies : Objective C, Java, MySQL, iOS, GlassFish Server.

- **Xcode**

Xcode is a suite of tools, developed by Apple, for developing software for Mac OS X and iOS. The main application of the suite is the integrated development environment (IDE), also named Xcode. The Xcode suite also includes most of Apple's developer documentation, and Interface Builder, an application used to construct graphical user interfaces. It supports C, C++, Objective-C, Objective-C++, Java, AppleScript, Python and Ruby source code with a variety of programming models

You can get it from here : <http://developer.apple.com/xcode/>

- **iPad Simulator**

This simulator provides environment of iOS to test our application. Simulator provides some special features like shake gesture, rotate iPad, etc.

- **Interface Builder**

Interface Builder is a software development application for Apple's Mac OS X operating system. Interface Builder allows Cocoa and Carbon developers to create interfaces for applications using a graphical user interface. The resulting interface is stored as a .nib file, short for NeXT Interface Builder, or more recently, as a .xib file. Interface Builder provides palettes, or collections, of user interface objects to an Objective-C developer. These user interface objects contain items like text fields, data tables, sliders, and pop-up menus. Interface Builder's palettes are completely extensible, meaning any developer can develop new objects and add palettes to Interface Builder.

To build an interface, a developer simply drags interface objects from the palette onto a window or menu. Actions (messages) which the objects can emit are connected to targets in the application's code and outlets (pointers) declared in the application's code is connected to specific objects. In this way all initialization is done before runtime, both improving performance and streamlining the development process.

- **Net Beans**

NetBeans refers to an integrated development environment (IDE) for developing with Java, JavaScript, PHP, Python, Ruby, Groovy, C, C++, Scala, Clojure, and others.

The NetBeans IDE is written in Java and can run anywhere a JVM is installed, including Windows, Mac OS, Linux, and Solaris. A JDK is required for Java development functionality.

- **Objective C**

Objective-C is a reflective, object-oriented programming language that adds Smalltalk-style messaging to the C programming language. Objective-C is a thin layer on top of C, and moreover is a strict superset of C; it is possible to compile any C program with an Objective-C compiler, and to freely include C code within an Objective-C class.

Objective-C derives its object syntax from Smalltalk. All of the syntax for non-object-oriented operations (including primitive variables, preprocessing, expressions, function declarations, and function calls) are identical to that of C, while the syntax for object-oriented features is an implementation of Smalltalk-style messaging. In Objective-C one does not call a method; one sends a message.

- **Java**

Java is a programming language originally developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities.

In our application Java is used for developing server web services, Which are used by our client i.e. iPad.

- **MySQL**

MySQL is a relational database management system (RDBMS)[1] that runs as a server providing multi-user access to a number of databases.

MySQL is used for storing different data like template details, user details, details of payment, etc...

- **iOS**

iOS (known as iPhone OS prior to June 2010) is Apple's mobile operating system. Originally developed for the iPhone, it has since been extended to support other Apple devices such as the iPod touch, iPad and Apple TV.

The user interface of iOS is based on the concept of direct manipulation, using multi-touch gestures. Interface control elements consist of sliders, switches, and buttons.

- **GlassFish Server**

For communication between client and server we are using GlassFish as server which is used to respond different SOAP request generated by client.

2.1 INTRODUCTION

Infostretch MobiMinted is a stand-alone application which allows user to design different cards for his/her life's special occasions like wedding, birthday party, baby birth announcements, graduation announcements etc. Infostretch MobiMinted utilizes various features of iPad as well as PayPal services for online payment.

MobiMinted is an online store offering modern custom stationery and photo cards for all of life's special events. MobiMinted carries exclusive modern designs that can't be found anywhere else, printed on thick, highest-quality paper, including 100% recycled paper, in a variety of innovative formats.

All cards as well as font details for respective card lie on server. Administrator can update database lying on server which contains all these details. Administrator can add new cards as well as their font details and labels' positions.

Objectives of the Project

The application should be user friendly in order to design and personalize cards. User should be able to pay online without much difficulty.

Project Profile:

Hardware Requirements:

Client Configuration: Apple iPad

Server Configuration:
CPU
Operating System: Windows XP/Vista/7
RAM: 2 GB
20 GB Minimum Space on Drive
2.00 dual GHz

2.2 FEATURES OF MOBIMINTED

MobiMinted contains many features which are mentioned below:

- **User:** Registration, Login
- **eCard Categories:** Add/Edit Categories or Sub-Categories
- **Template:** Add/Edit Image, Paper Type and Quantity
- **eCards:** View/Edit
- **Customize eCard from iPad:** Label, Change Text, Zooming
- **Customization options from Server:** Color, Number of Labels, Position of Label, Label component size
- **Sample:** The user can get a Sample eCard for selected template
- **Payment:** PayPal Sandbox Integration

3.1 REQUIREMENT OF THE SYSTEM

As MobiMinted is developed as an independent application for iPad and not as a client centric project, the requirements are not so clear as they are in any client centric project.

- **Functional Requirements for MobiMinted**

R1: Registration

Description:

Registration function determines the availability of e-mail address entered by user. If e-mail address is not available then it shows alert message to user to register with different e-mail address. Otherwise it shows message to user that his/her account has been created successfully.

Input:

User should provide User Name, E-mail address and Password

Output:

If user enters invalid e-mail address, alert message should be displayed.
If user enters valid e-mail address, user can view his/her previously saved designs as well as user is able to pay online using PayPal.

Processing:

As user enters e-mail address and password, client side validation and server side validation are carried out.

Table: User_info

USERNAME	VARCHAR(45)
EMAILID	VARCHAR(100)
PASSWORD	VARCHAR(45)

Table 3.1 User_info table

R2: Login

Description:

Login function determines the correctness of combination of e-mail address and password. If both e-mail address and password are found to be correct then user is allowed to proceed further. Otherwise alert message is displayed to enter correct credentials.

Input:

User should provide E-mail address and Password

Output:

If user enters invalid e-mail address/password combination, alert message notifying user that he has entered wrong combination should be displayed otherwise user should be able to proceed ahead into the application.

Processing:

As user enters e-mail address and password, client side validation and server side validation are carried out.

Table: User_info

Email ID	VARCHAR(100)
password	VARCHAR(50)

Table 3.2 User_info table

R3: Choose Category

Description:

Application should provide facility to choose a category from different categories (e.g. Wedding, Party, Baby & Kids, Calendars, Holiday, and Stationary).

Input:

User should select one of the categories (e.g. Wedding, Party, Calendars)

Output:

As soon as user selects one of the categories, he/she should be given another selection for subcategory.

Table: Category

CATEGORYID	INT
CATEGORYNAME	VARCHAR(45)

Table 3.3 Category table

R3.1: Choose Subcategory**Description:**

Application should provide facility to choose a subcategory from different subcategories

Input:

User should select one of the subcategories

Output:

As soon as user selects subcategory, templates of selected subcategory should be displayed so that user can view/personalize that templates.

Processing:

Templates are fetched from server and displayed

Table: Subcategory

CATEGORYID	INT
SUBCATEGORYID	INT
SUBCATEGORYNAME	VARCHAR(45)

Table 3.4 Subcategory table

R4: Personalize a Card

Description:

Personalize a card is the core facility of this project. By personalizing a card one can represent one's thoughts on a card. Personalize function makes user personalize selected card by editing text labels; selecting quantity, paper type, template color.

R4.1: Edit text label

Description:

User should be able to write on cards or edit text written on card.

Input:

User should write into text fields provided for labels

Output:

Preview of template showing text labels

R4.2: Select Quantity

Description:

User should be given choice to select quantity as per needed by user

Input:

Selection of quantity of cards

Output:

Quantity is added into user's cart

R4.3: Select paper type

Description:

User should be given chance to select the type of paper for printing the card

Input:

Selection of paper type

Output:

Paper type is added into user's cart

R4.4: Save Design

Description:

It might be the case that user just wants to save the design of card for time being and edit later on. So user should be able to save design of the card.

Input:

User should tap on "Save Design" button

Output:

User's design will be saved into user's cart

R4.5: View Saved Design

Description:

If user wants to see which are the designs that he has saved so that he/she can edit those designs later on.

Input:

User should tap on "View Saved Designs" button

Output:

All the saved design cards will be displayed to user so that he/she will be able to edit those cards.

Table: Edit_Template

EDITTEMPLATEID	INT
EMAILID	VARCHAR(100)
TEMPLATEID	INT
LABEL	TEXT
LABEL_X	INT
LABEL_Y	INT

Table 3.5 Edit_Template table

R5: Add to Sample card**Description:**

If user wants to experience type of quality of paper or wants to see how his/her would card looking like before ordering into bulk of cards, user is provided this facility to add the card into sample. By doing so, user has to pay a certain amount for the card that includes shipping cost.

Input:

User should tap on "Add to Sample" button

Output:

Sample card is added to user's cart

R6: Zooming effect**Description:**

There are cases when user writes on card and he/she won't be able to read in simulator due to small font size. In these cases user should be given facility like zooming effect so that he/she will be able to read clearly.

Input:

User has to tap on Zoom in button

Output:

As soon as user taps on zoom in button, an enlarged image is displayed to the user.

R7: Online Payment Gateway**Description:**

By using online payment gateway user can pay for selected card. PayPal is used as online payment gateway. User must be having an account on PayPal. When user creates an account on PayPal he will be given user id and password that he has to use while paying using PayPal.

Input:

User has to provide user id and password given by PayPal

Output:

If there is enough balance in account, successful message notifying user that payment is done is displayed otherwise user will be given alert message that there is not enough money in account.

R8: Logout**Description:**

User has a wide range of functionalities that he can use while he/she is logged in, but once he/she is logged out he/she has limited functionalities (i.e. user can only view the cards)

Input:

User should tap on "Logout" button

Output:

User is logged out

- **Interface Requirements**

User Interface

- Touch screen Interface of Apple iPad

Hardware Interface

- Apple iPad

Software Interface

- Netbeans 6.x
- MySQL
- Apple iOS
- Mac OS X
- iPad Simulator

- **Performance Requirements**

Since MobiMinted uses web services lying on server, the network communication should be faster. Server should be fast enough to respond to user's requests.

3.2 USE CASE DIAGRAMS

➤ Use Case Diagram for User of MobiMinted

Use case diagram for the whole project which defines the various functionalities provided to any user of MobiMinted

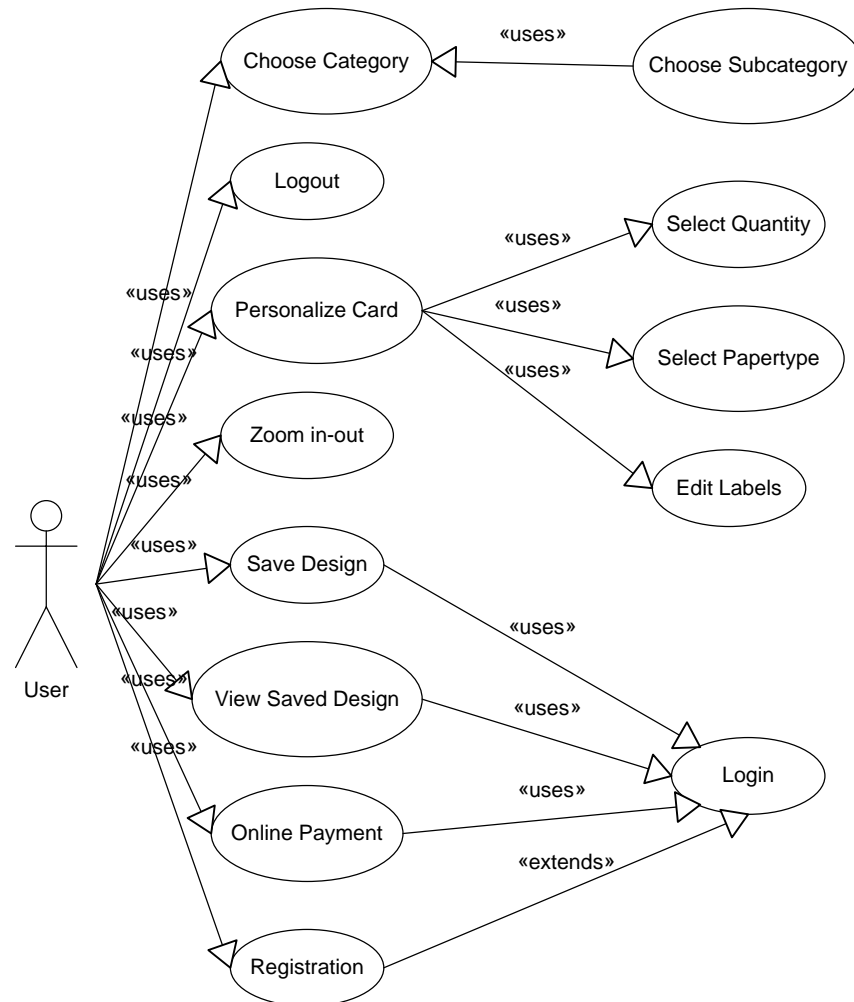


Figure 3.1 MobiMinted Usecase

➤ **Use Case Diagram for Administrator of MobiMinted**

Use case diagram which defines the roles of administrator

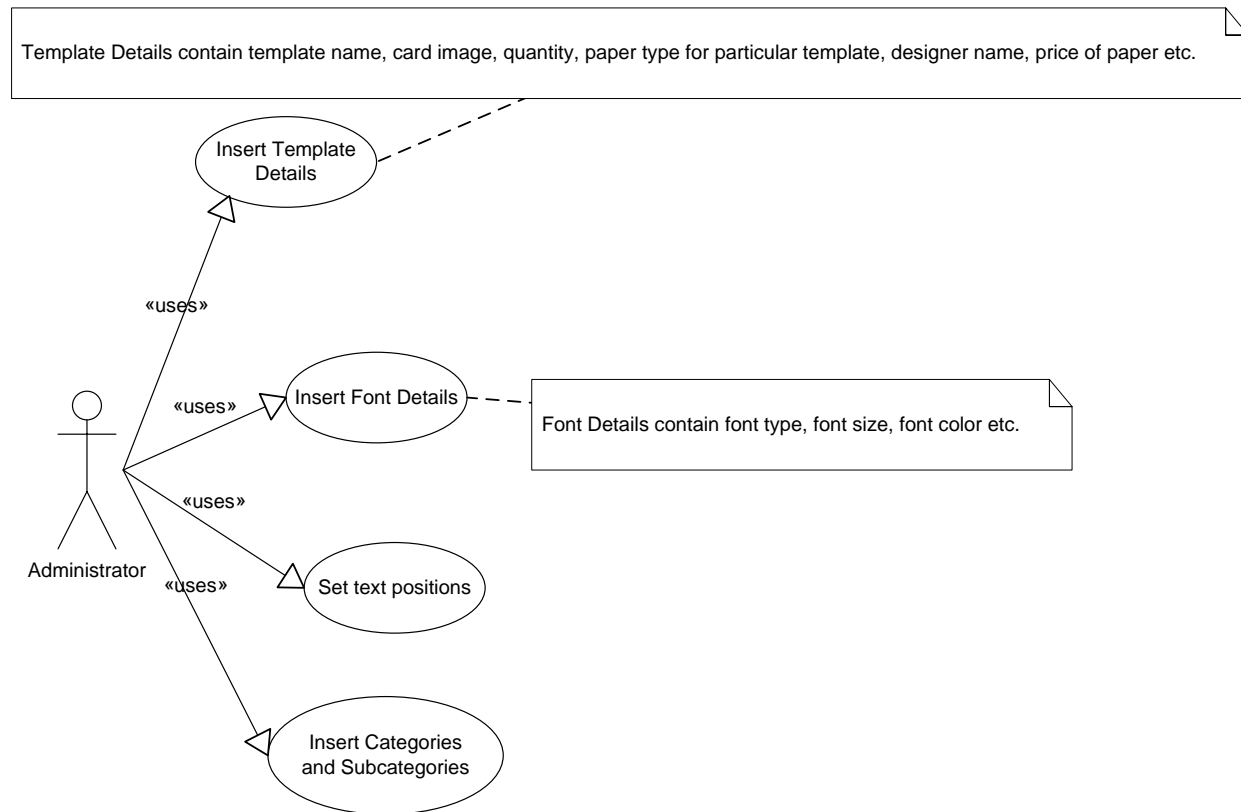


Figure 3.2 Administrator's Usecase

3.3 SEQUENCE DIAGRAMS

➤ **Sequence diagram for viewing templates of different categories and subcategories**

Sequence diagram for viewing templates defines the flow in which templates are displayed on iPad

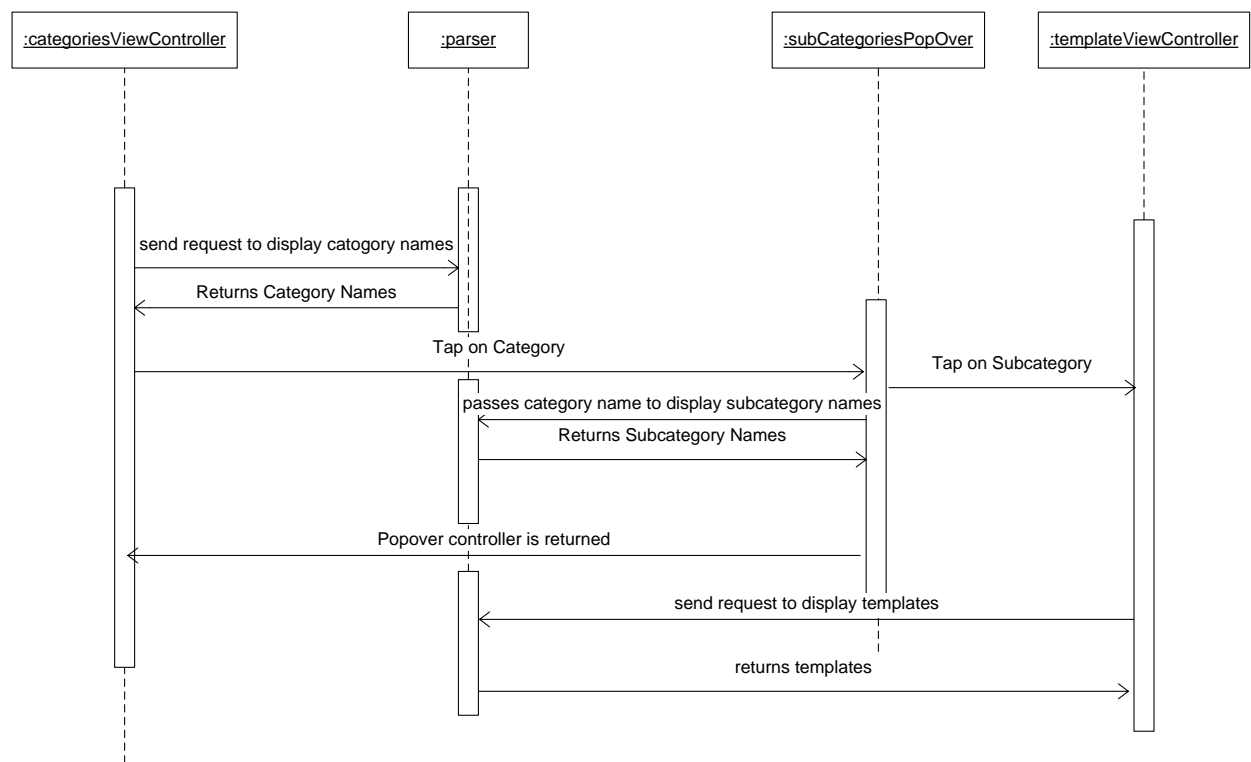


Figure 3.3 Templates Sequence Diagram

➤ **Sequence diagram for saving design and retrieving saved designs**

Sequence diagram for saving design and retrieving saved designs defines the flow of saving design and retrieving back saved designs.

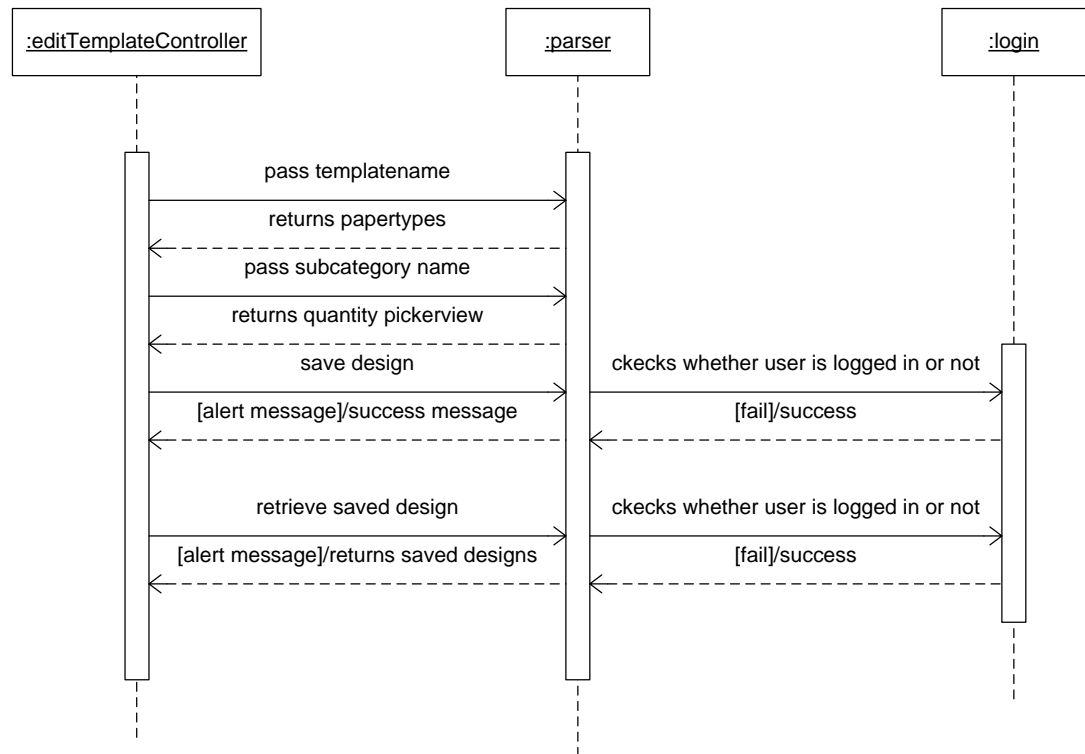


Figure 3.4 save design and view saved design sequence diagram

3.4 ACTIVITY DIAGRAMS

➤ **Activity diagram for personalize a card**

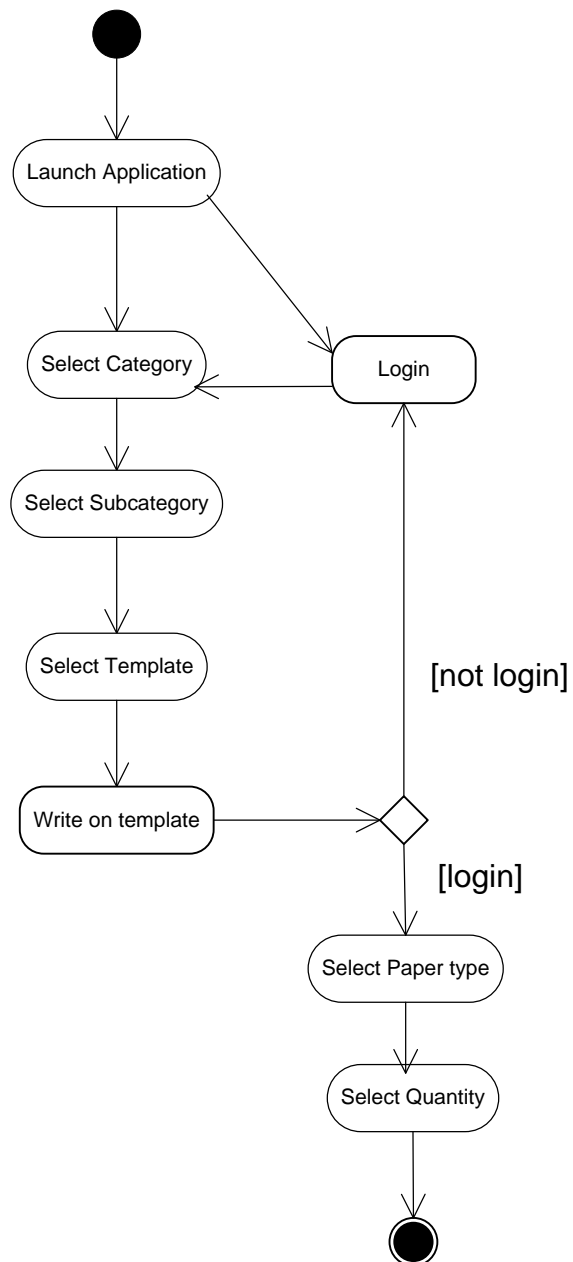


Figure 3.5 Personalize card activity diagram

➤ **Activity diagram for saving design**

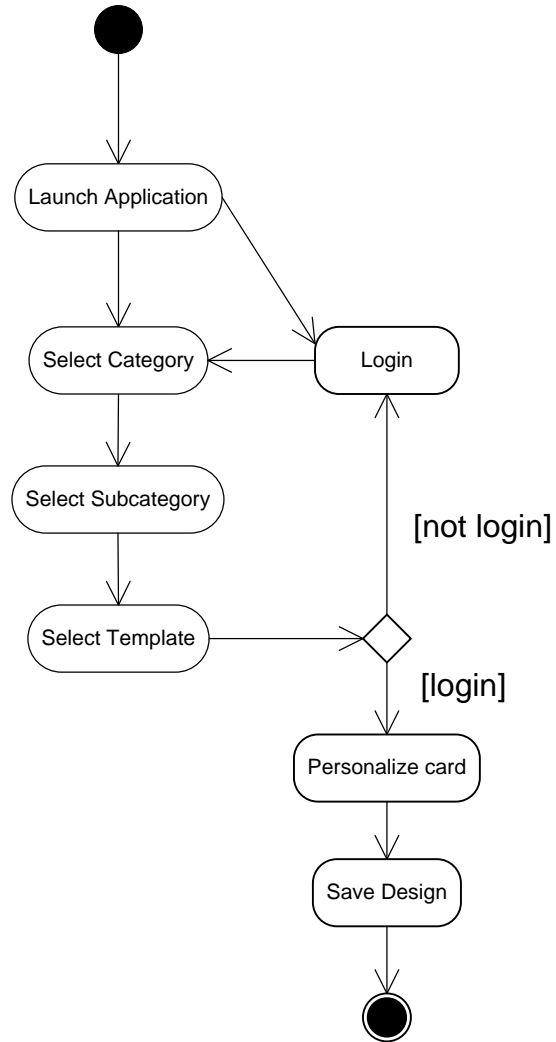


Figure 3.6 Save design activity diagram

➤ **Activity diagram for viewing saved design**

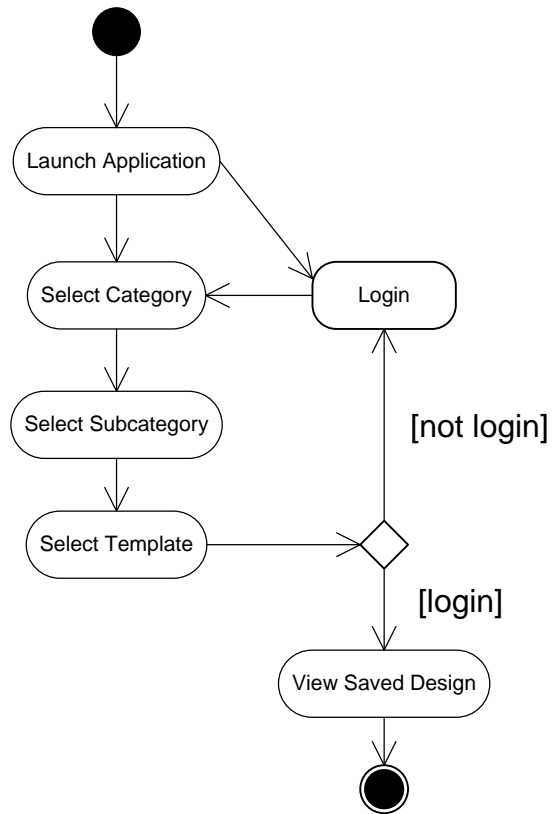


Figure 3.7 View saved design activity diagram

3.5 CLASS DIAGRAMS

➤ Class diagram for MobiMinted

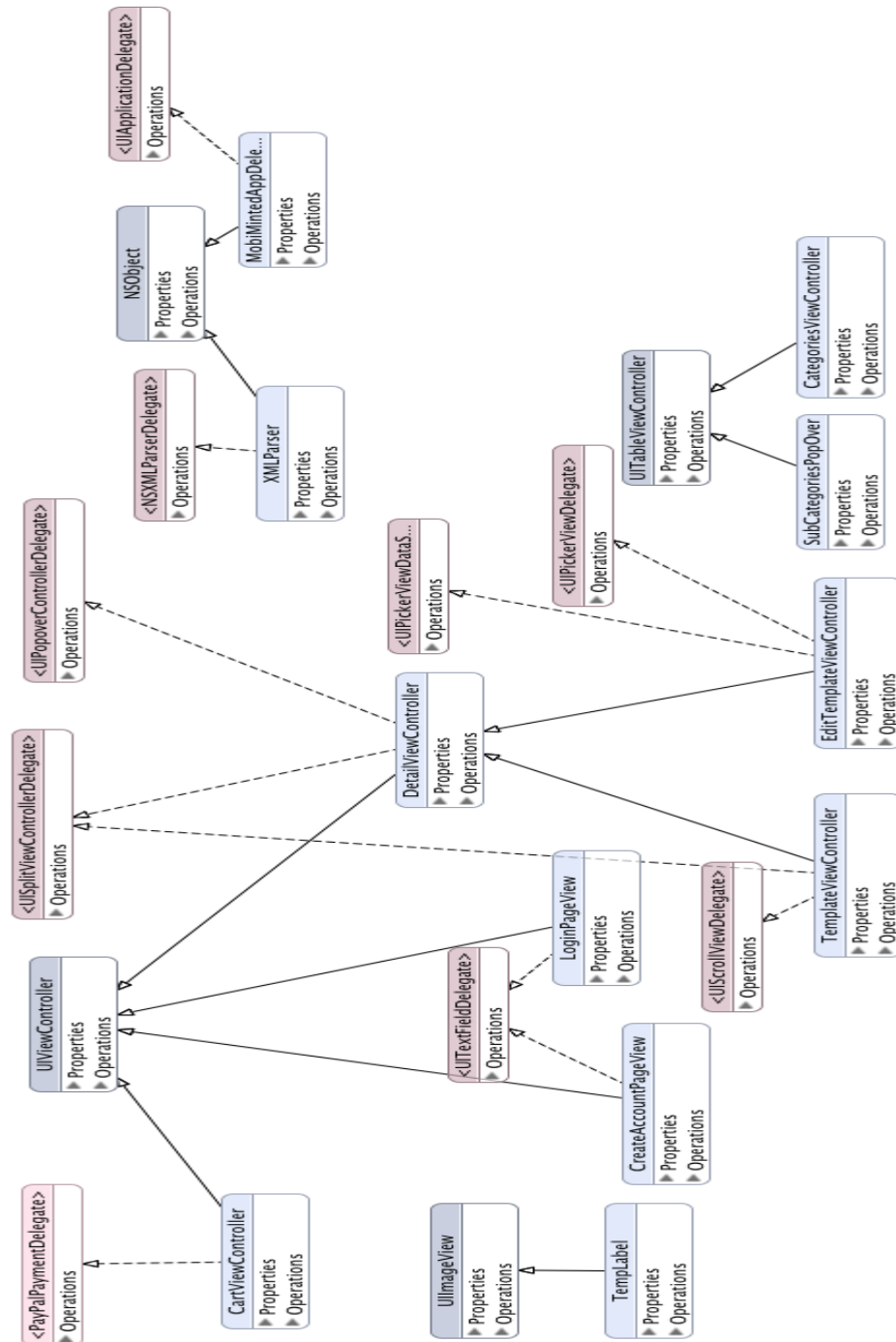


Figure 3.8 MobiMinted Class diagram

➤ **MobiMintedAppDelegate**

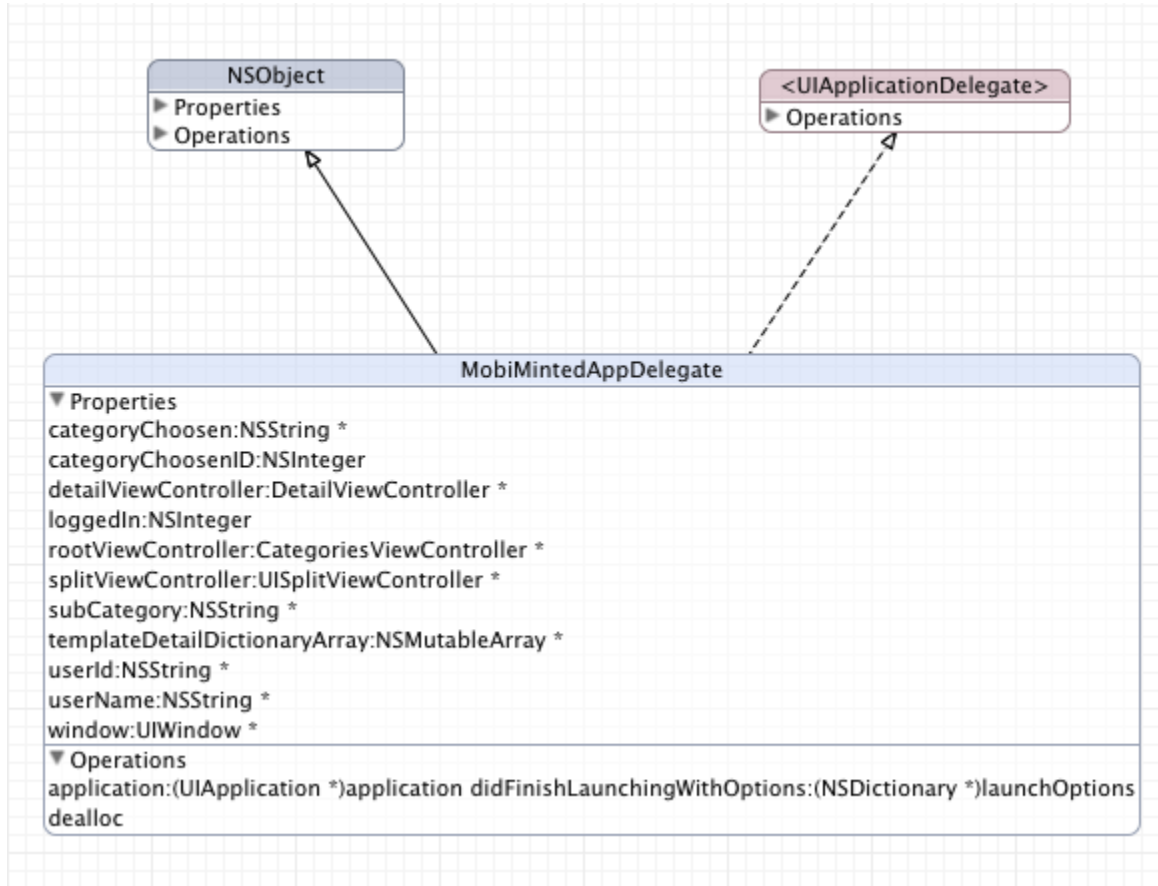


Figure 3.9 MobiMintedAppDelegate Class diagram

➤ **CategoriesViewController Class diagram**

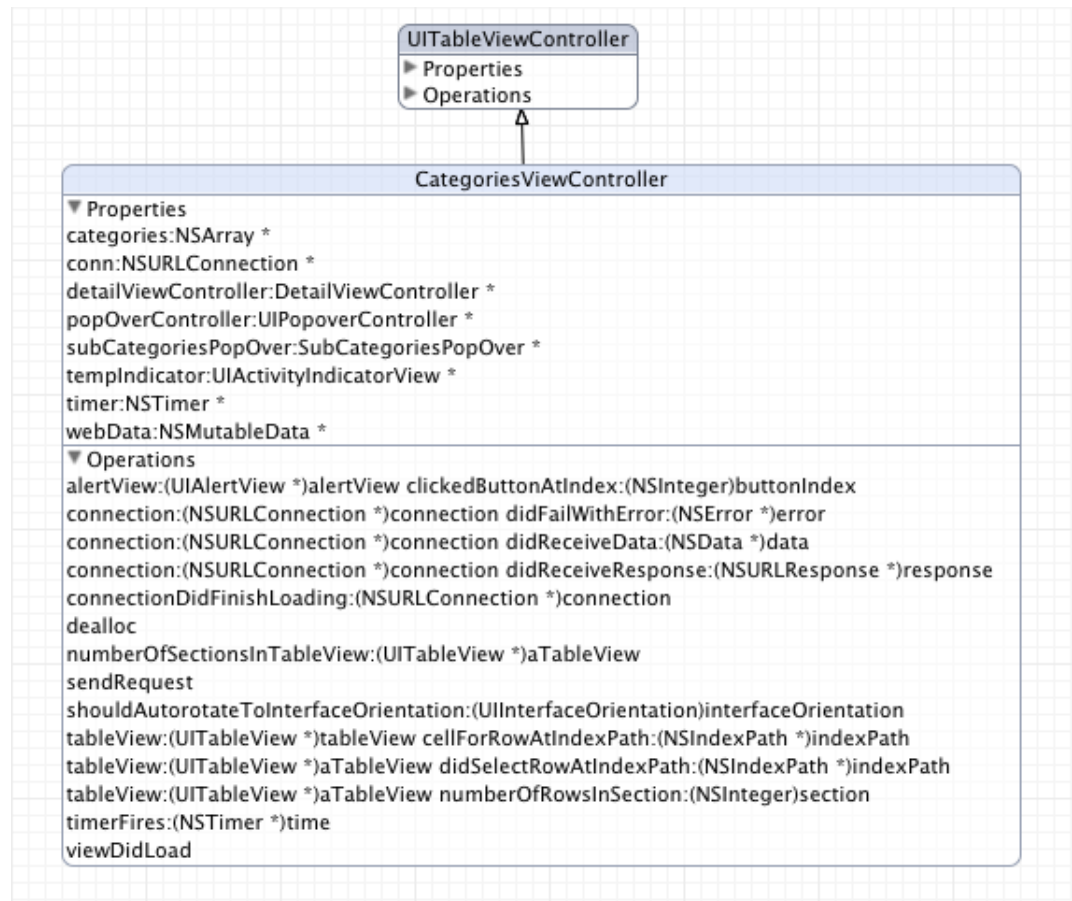


Figure 3.10 CategoriesViewController Class diagram

➤ **SubcategoriesViewController Class diagram**

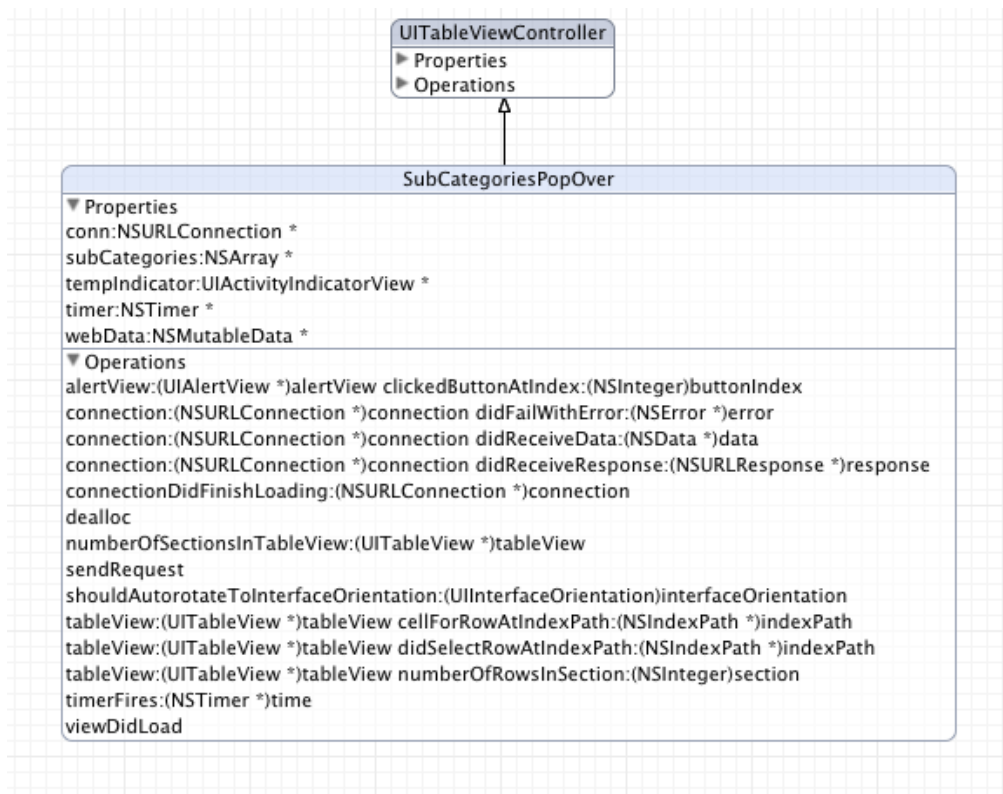


Figure 3.11 SubcategoriesViewController Class diagram

➤ DetailViewController Class diagram

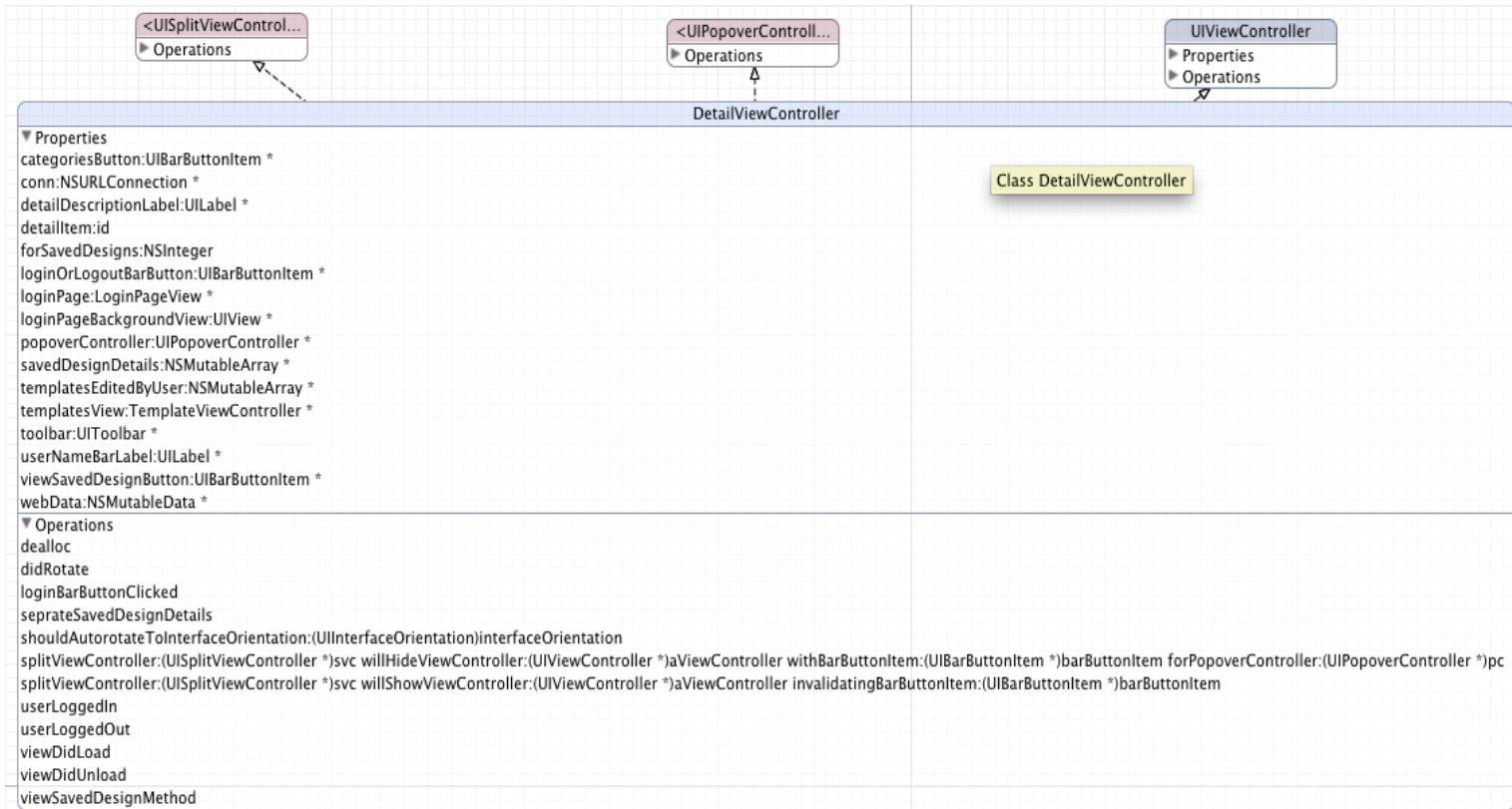


Figure 3.12 DetailViewController Class diagram

➤ CartViewController Class diagram

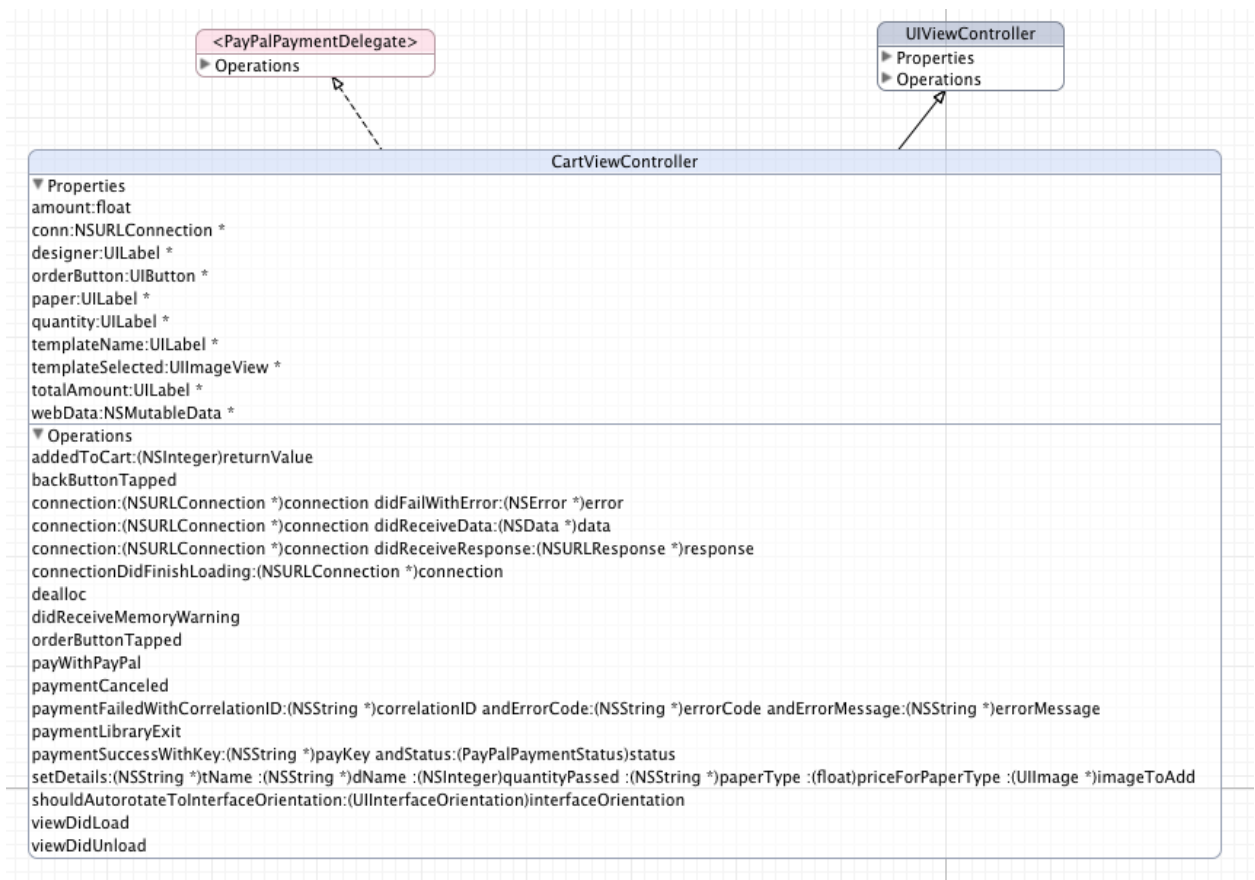


Figure 3.13 CartViewController Class diagram

➤ **LoginPageView Class diagram**

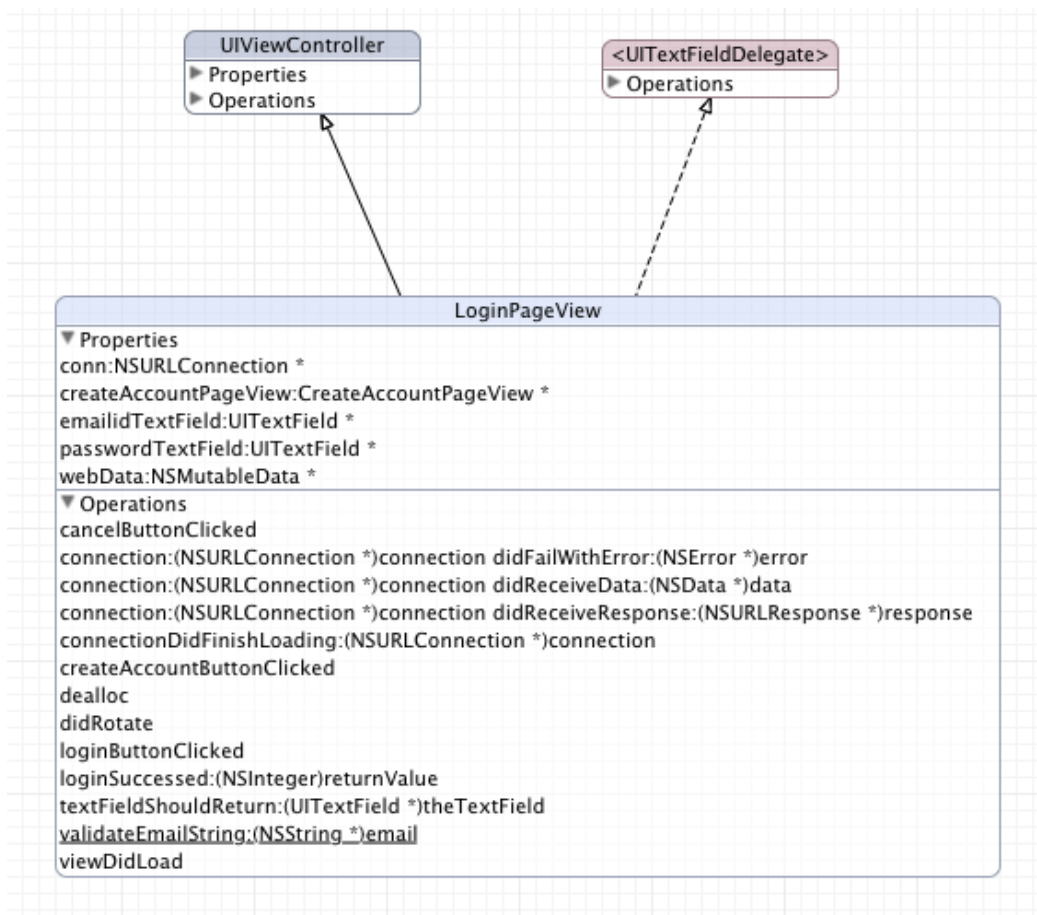


Figure 3.14 LoginPageView Class diagram

➤ **CreateAccountPageView Class diagram**

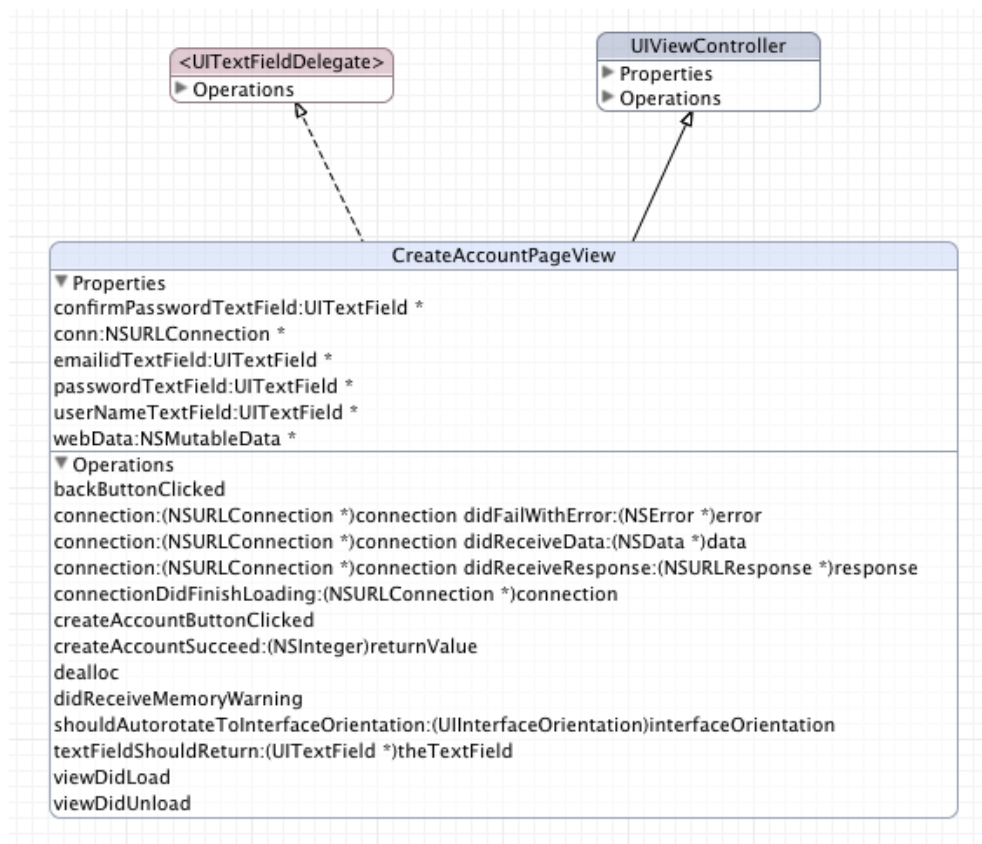


Figure 3.15 CreateAccountPageView Class diagram

➤ XMLParser Class diagram

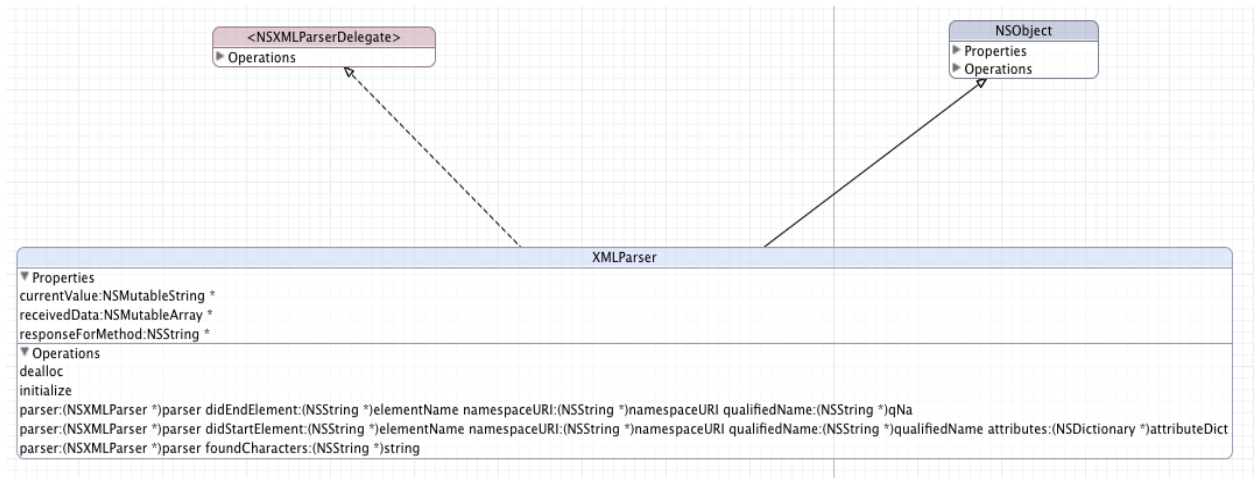


Figure 3.16 XMLParser Class diagram

➤ TempLabel Class diagram

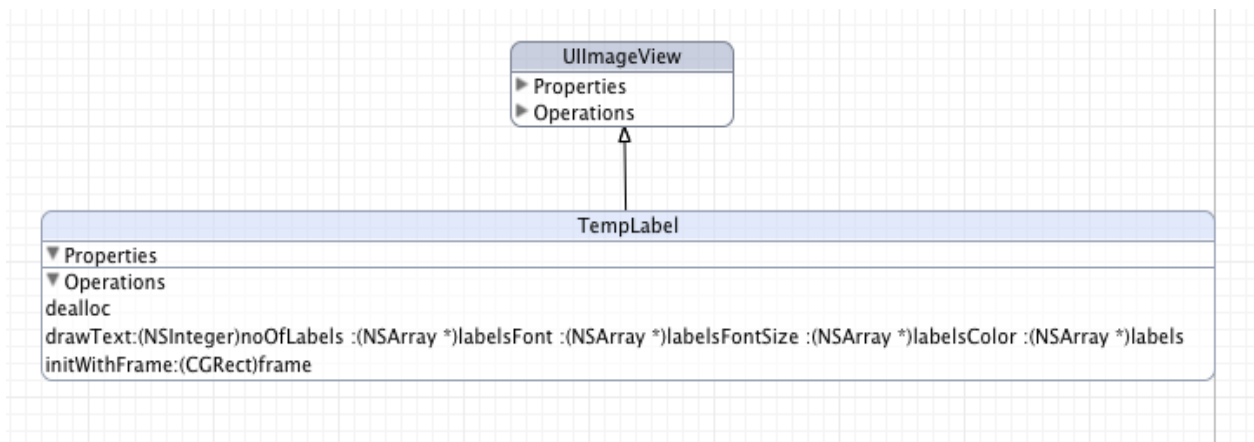


Figure 3.17 TempLabel Class diagram

➤ EditTemplateViewControllerProperties

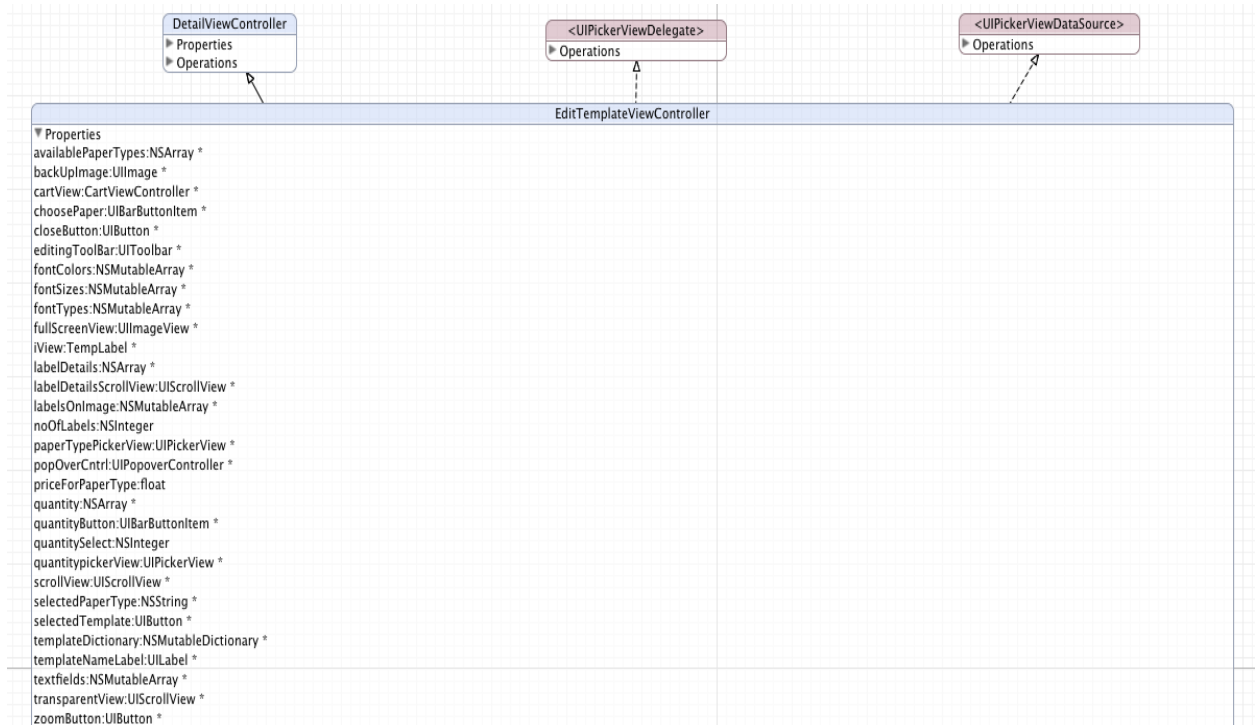


Figure 3.18 EditTemplateViewControllerProperties

➤ EditTemplateViewControllerActions

▼ Operations
OKButtonTapped
cancelButtonTapped
cancelPaperType
choosePaperTapped
closeButton:(id)sender
connection:(NSURLConnection *)connection didFailWithError:(NSError *)error
connection:(NSURLConnection *)connection didReceiveData:(NSData *)data
connection:(NSURLConnection *)connection didReceiveResponse:(NSURLResponse *)response
connectionDidFinishLoading:(NSURLConnection *)connection
dealloc
didReceiveMemoryWarning
didRotateEditTemplateViewController
dispPaperTypes
displayActionSheet
getFontTypesandColor
initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
keyboardHidden:(NSNotification *)aNotification
keyboardShown:(NSNotification *)aNotification
nextButton
numberOfComponentsInPickerView:(UIPickerView *)pickerView
paperTypeSelected
pickerView:(UIPickerView *)pickerView didSelectRow:(NSInteger)row inComponent:(NSInteger)component
pickerView:(UIPickerView *)pickerView numberOfRowsInComponent:(NSInteger)component
pickerView:(UIPickerView *)pickerView viewForRow:(NSInteger)row forComponent:(NSInteger)component reusingView:(UIView *)view
quantityButtonTapped
quantitySelected
saveDesignTapped
separateDetails
showCategoriesEditTemplateViewController
splitViewController:(UISplitViewController *)svc willHideViewController:(UIViewController *)aViewController withBarButtonItem:(UIBarButtonItem *)barButtonItem forPopoverController:(UIPopoverController *)pc
splitViewController:(UISplitViewController *)svc willShowViewController:(UIViewController *)aViewController invalidatingBarButtonItem:(UIBarButtonItem *)barButtonItem
templateSelected:(id)sender :(NSMutableArray *)arrayToDisplay
textFieldShouldReturn:(UITextField *)textField
viewDidLoad
zoomButtonTapped

Figure 3.19 EditTemplateViewControllerActions

➤ TemplateViewControllerProperties

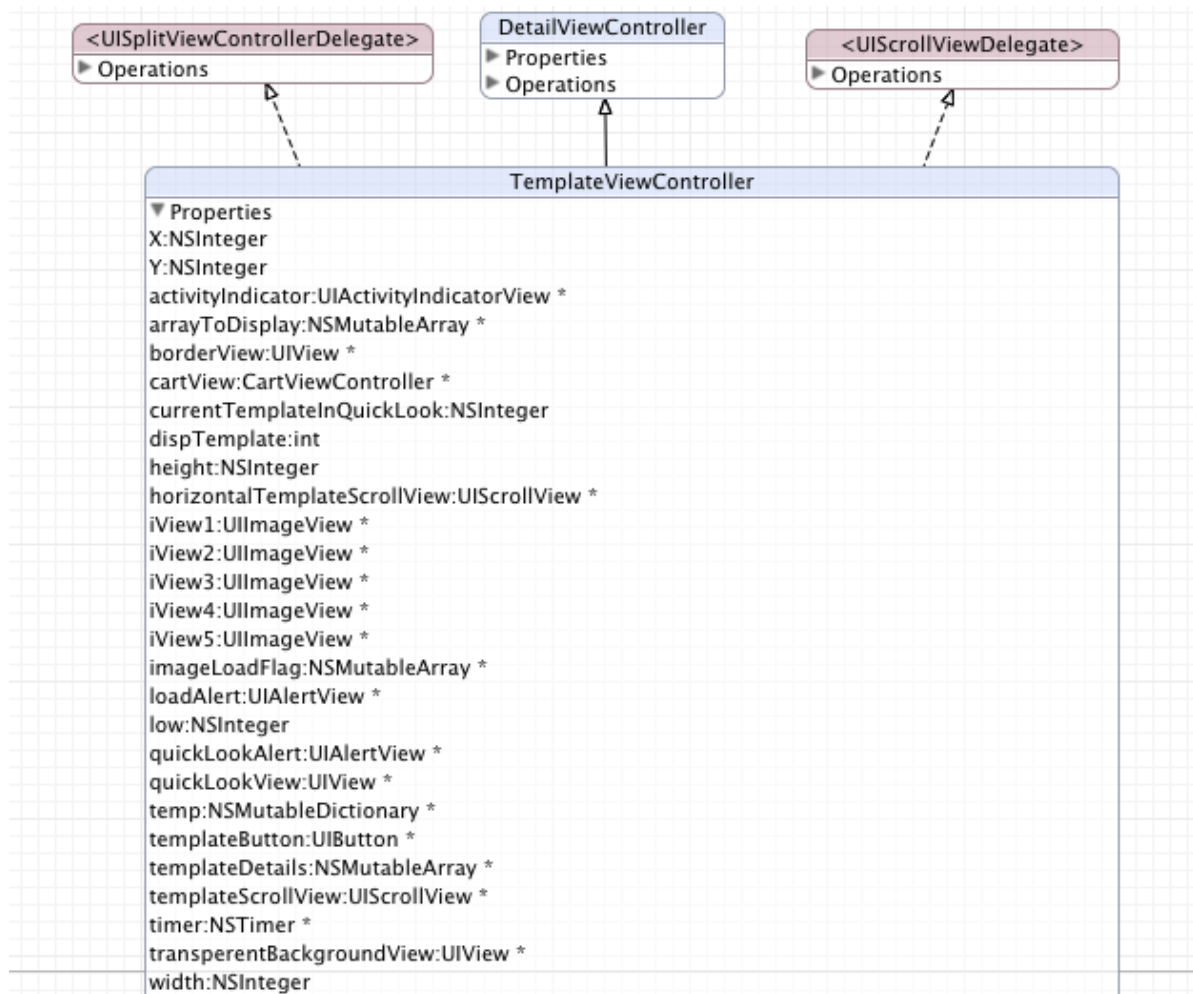
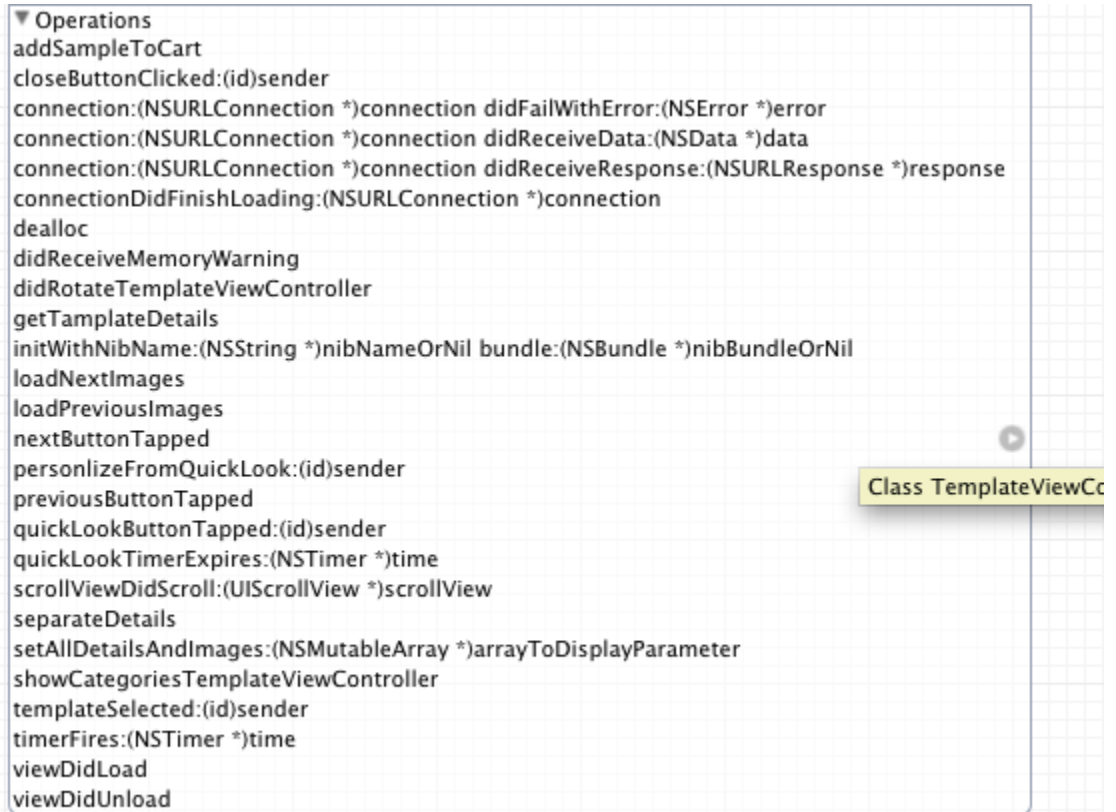


Figure 3.20 TemplateViewControllerProperties

➤ **TemplateViewControllerActions**



The screenshot shows the 'Operations' section of the `TemplateViewController` class in Xcode. The list of methods includes various actions related to data fetching, UI updates, and lifecycle events. A yellow tooltip on the right side of the list reads 'Class TemplateViewCo'.

```
▼ Operations
addSampleToCart
closeButtonClicked:(id)sender
connection:(NSURLConnection *)connection didFailWithError:(NSError *)error
connection:(NSURLConnection *)connection didReceiveData:(NSData *)data
connection:(NSURLConnection *)connection didReceiveResponse:(NSURLResponse *)response
connectionDidFinishLoading:(NSURLConnection *)connection
dealloc
didReceiveMemoryWarning
didRotateTemplateViewController
getTemplateDetails
initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
loadNextImages
loadPreviousImages
nextButtonTapped
personalizeFromQuickLook:(id)sender
previousButtonTapped
quickLookButtonTapped:(id)sender
quickLookTimerExpires:(NSTimer *)time
scrollViewDidScroll:(UIScrollView *)scrollView
separateDetails
setAllDetailsAndImages:(NSMutableArray *)arrayToDisplayParameter
showCategoriesTemplateViewController
templateSelected:(id)sender
timerFires:(NSTimer *)time
viewDidLoad
viewDidUnload
```

Figure 3.21 TemplateViewControllerActions

3.6 PROJECT MANAGEMENT

➤ **Feasibility Study**

Technical Feasibility

Keeping in mind about existing technology, technical support is provided to hold data. Infostretch MobiMinted is feasible since it is developed using Objective C.

Operational Feasibility

Whether application will be used if it is developed and implemented or there will be resistance in using it by user it may undermine the possible application benefits. There are not many barriers in developing Infostretch MobiMinted.

Time Scheduling Feasibility

Projects are initiated with specific deadlines. We need to evaluate whether the deadlines are mandatory or desirable. Time is the one of the critical factor in the development of any system but this kind of feasibility is hardly perfect in any system. Infostretch MobiMinted is supposed to be completed within four months of industrial training. It is feasible to develop Infostretch MobiMintred within this span of period.

Implementation Feasibility

A proper implementation is essential to provide a reliable system meet the requirement of the organization. Implementation is the stage in the project where the theoretical design is turned into a working system. The most critical stage in achieving a new

successful system is to improve the performance of the existing system and to proposed system effective.

4.1 DATA DICTIONARY

➤ **Data Dictionary for category**

Category			
Name	Type	Nullable	Description
categoryid	INT(20)	N	Primary Key of the table
categoryname	VARCHAR(50)	N	Defines the name of the category

Table 4.1 Table for category

➤ **Data Dictionary for subcategory**

Subcategory			
Name	Type	Nullable	Description
subcategoryid	INT(20)	N	Primary Key of the table
categoryid	INT(20)	N	Describes the foreign key to the table category
subcategoryname	VARCHAR(50)	N	Defines the name of the subcategory

Table 4.2 Table for subcategory

➤ **Data Dictionary for templates**

Templates			
Name	Type	Nullable	Description
templateid	INT(20)	N	Primary Key of the table
categoryid	INT(20)	N	Defines the id of the category
subcategoryid	INT(20)	N	Describes the foreign key to the table subcategory
templatename	VARCHAR(200)	N	Describes the name of the

			template
templatecolor	VARCHAR(25)	Y	Describes the colour of the template
designer	VARCHAR(50)	Y	Describes the name of the designer of the card
imageName	VARCHAR(40)	N	Describes the name of the card's image

Table 4.3 Table for templates

➤ **Data Dictionary for labeldetails**

Labeldetails			
Name	Type	Nullable	Description
labeldetailid	INT(20)	N	Primary Key of the table
templateid	INT(20)	N	Defines the foreign key to the table templates
labeldata	VARCHAR(500)	N	Describes the label to be printed on card
label_X	INT(20)	N	Describes the position of the label
label_Y	INT(20)	N	Describes the position of the label
labelColor	VARCHAR(25)	N	Describes the colour of the font
labelFontSize	INT(11)	N	Describes the size of the font
labelFont	VARCHAR(50)	N	Describes the font of the label

Table 4.4 Table for labeldetails

➤ **Data Dictionary for papertype**

Papertype			
Name	Type	Nullable	Description
papertypeid	INT(20)	N	Primary Key of the table
templateid	INT(20)	N	Defines the foreign key to the table templates
Papertype	VARCHAR(100)	N	Describes the type of the paper
Price	FLOAT(11)	N	Describes the price of the paper

Table 4.5 Table for papertype

➤ **Data Dictionary for usercart**

Usercart			
Name	Type	Nullable	Description
emailid	VARCHAR(100)	N	Describes the email address of the user
templatename	VARCHAR(200)	N	Describes the name of the template
papertype	VARCHAR(100)	N	Describes the type of the paper
price	FLOAT(11)	N	Describes the price of the paper
quantity	INT(11)	N	Describes the quantity of the cards

Table 4.6 Table for usercart

➤ **Data Dictionary for user_info**

user_info			
Name	Type	Nullable	Description
Id	INT(11)	N	Primary Key of the table
emialid	VARCHAR(100)	N	Describes the email address of the user
Password	VARCHAR(100)	N	Describes the password of user's account
Username	VARCHAR(100)	N	Describes the name of the user

Table 4.7 Table for user_info

4.2 INTERFACE DESIGN

- **Interface Design for Landscape mode**

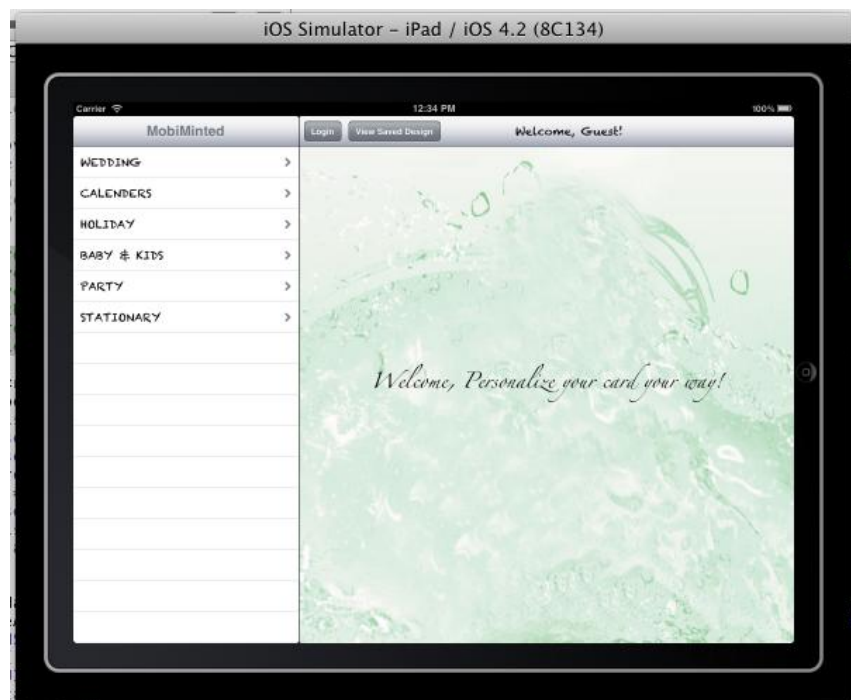


Figure 4.1 Interface Landscape mode

- **Interface Design for templates**

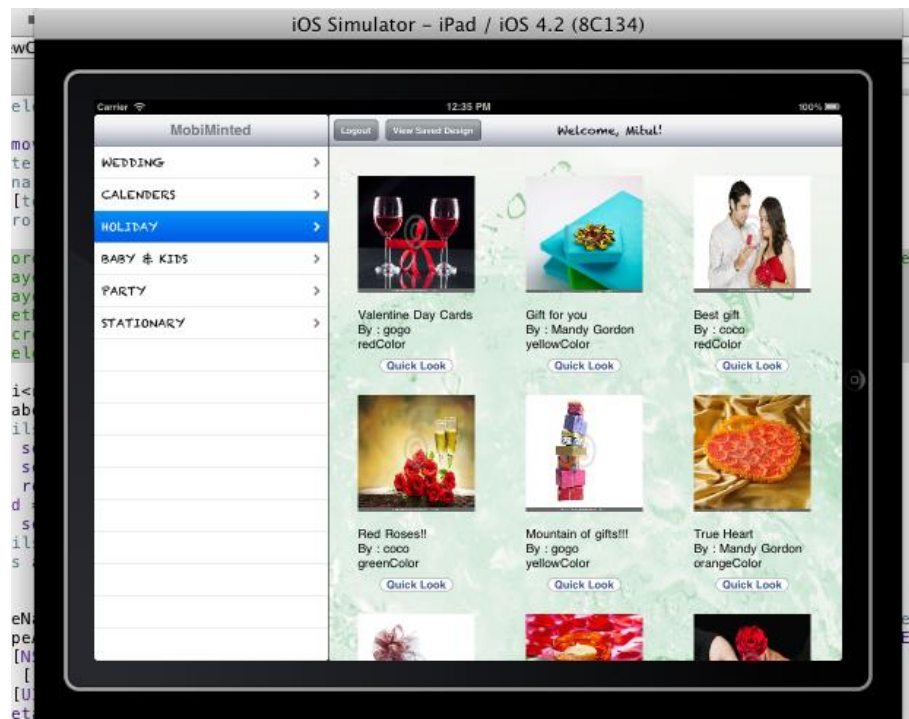


Figure 4.2 Interface templates view

- **Interface Design for Quick look view**

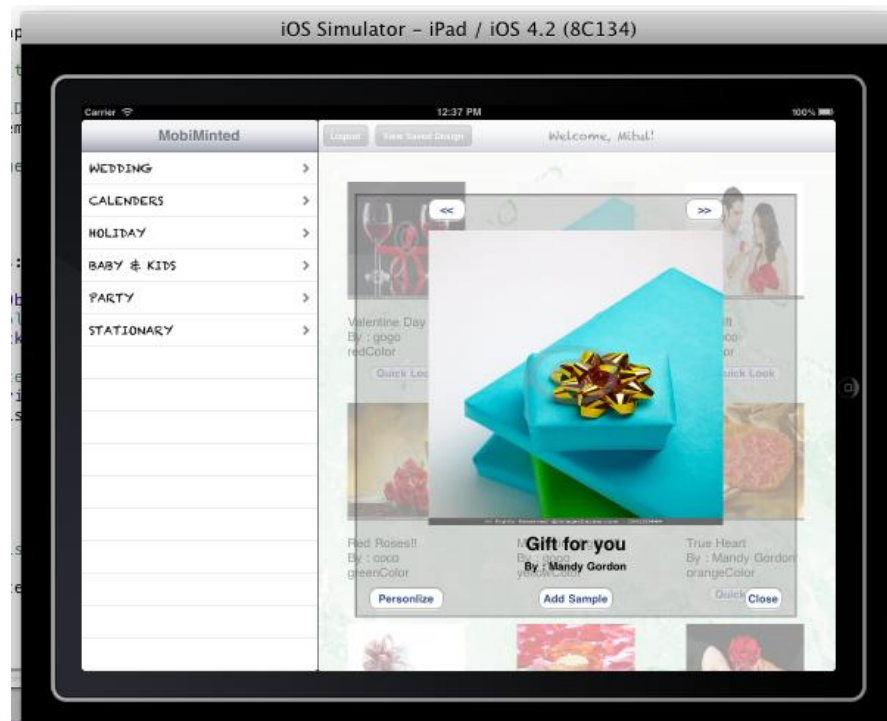


Figure 4.3 Interface Quick look view

- **Interface Design for Cart view**

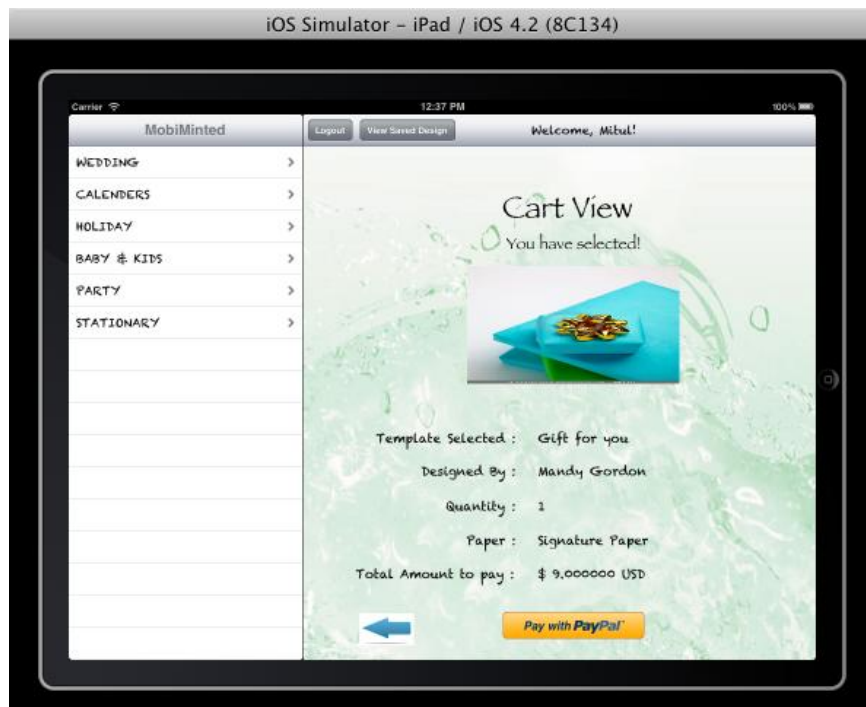


Figure 4.4 Interface Cart view

- **Interface Design for edit_template**



Figure 4.5 Interface personalize template

- **Interface Design for Designed Template**



Figure 4.6 Interface Designed Template

- **Interface Design for Save Design**



Figure 4.7 Interface Save Design

- **Interface Design for View Saved Design**

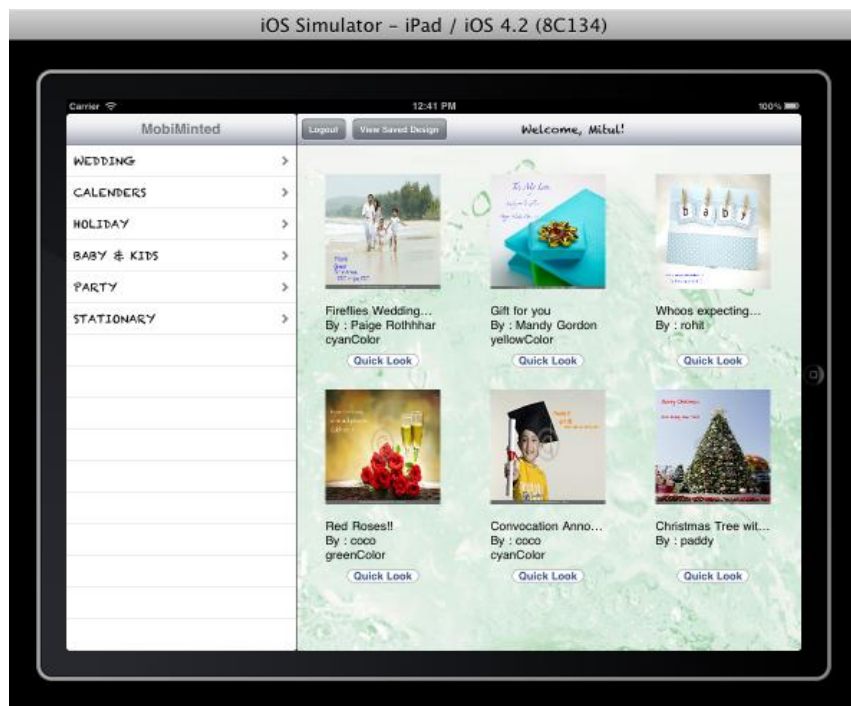


Figure 4.8 Interface Design View Saved Design

- **Interface Design for Payment Gateway**

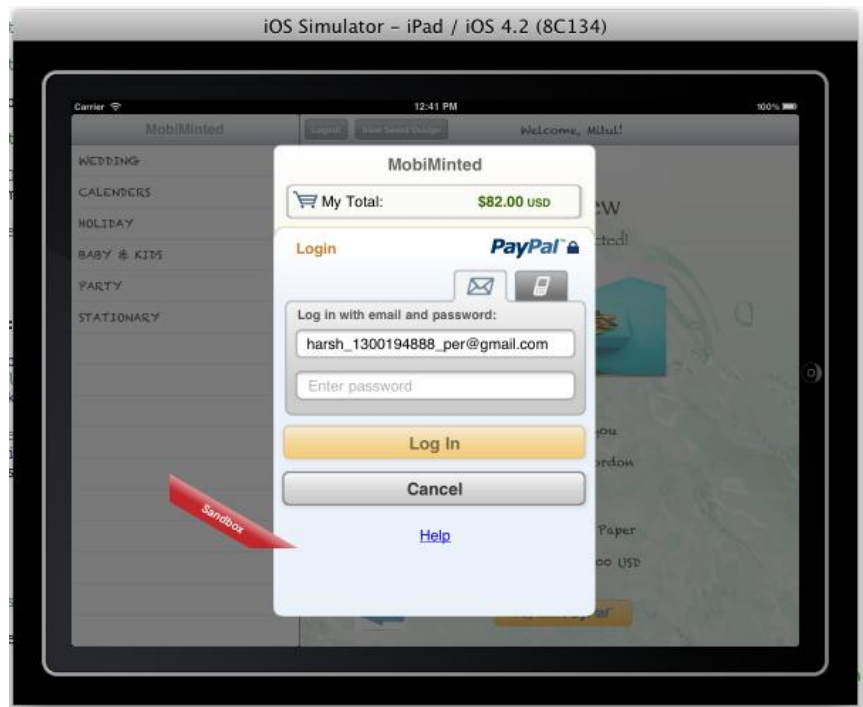


Figure 4.9 Interface Payment Gateway

4.3 VALIDATIONS

The validations involved in the application are categorized in following categories:

- Client Side Validations
- Server Side Validations

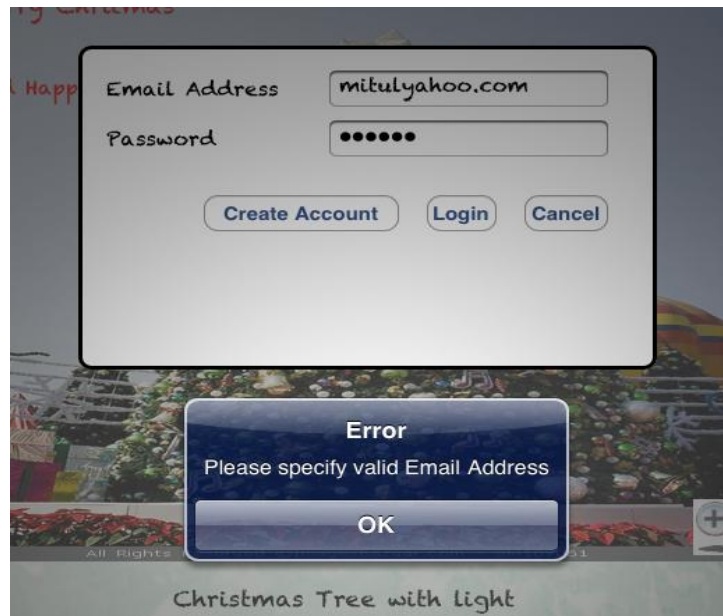


Figure 4.10 Client side validation

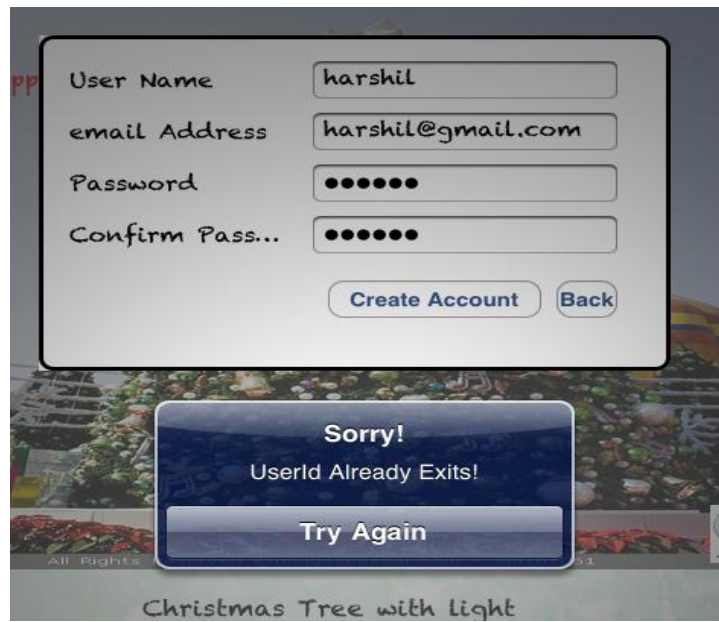


Figure 4.11 Server side validation

This is a single user system and basically uses touch interaction to perform different operations as that is the only way to interact with iPad. Here we are using Object Oriented approach for development of application as Objective C is an Object Oriented language. There are different classes or modules which are designed to perform some specified operations.

The different modules of this system are:

1. Application delegate
2. Categories View Controller
3. Detail View Controller
4. Subcategories View Controller
5. Login Page
6. Create Account
7. Template View Controller
8. Edit Template View Controller
9. Cart View Controller
10. XML Parser
11. Temp Label

- **Application delegate**

This class is automatically generated by system and is like universal class. We can get reference of this class in any other class and access its variables or methods. It also contains methods like `appDidFinishLaunching` which are being called by iOS first when your app gets started. In our app this class provides function to load root view and detail view when application is loaded and also provides some space for global variables.

- **Categories View Controller**

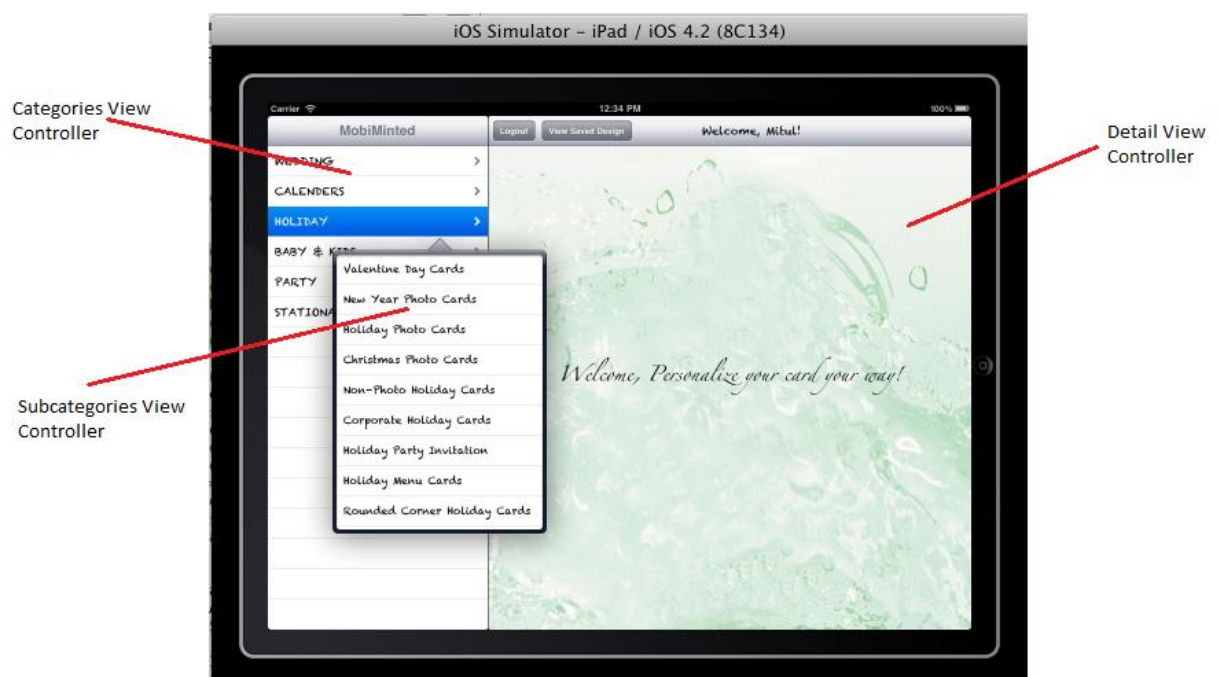
This class is for giving details of different categories available for cart according to occasion like Valentine Day Cards, Holiday, Wedding Cards, etc.... This view contains a table view to display diff categories. It first sends a request for list and responds for tap by calling method didSelectRowAtIndexPath:. It also stores the category chosen by user in a variable in appdel for further use like for loading subcategories which are dependent on categories.

- **Detail View Controller**

This class is super class of TemplateViewController and EditTemplateViewController and is on right side of screen. This class provides common operation that are supported by whole app like login, view previously saved design.

- **Subcategories View Controller**

This controller is displayed in a popover controller and contains a table view to display different subcategories for chosen categories, it prepares a SOAP request accordingly. And once user selects any of subcategory the application will load Template View Controller which is used to display different templates for that subcategory.



- **Login Page**

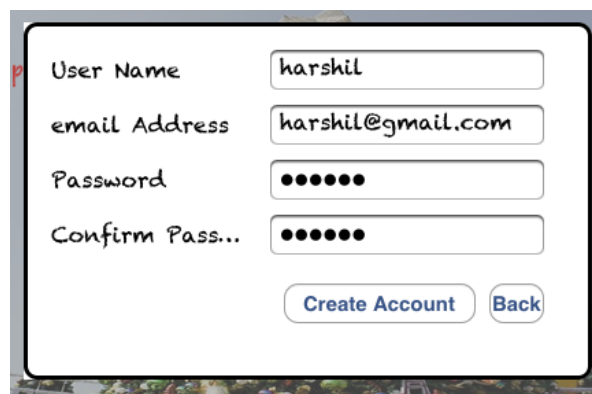
This class as name suggests provides features of login it takes input as username and password and makes a SOAP request to check authentication of user. This class also has a function which checks for validity of given email address.



The screenshot shows a login dialog box with a white background and a black border. It contains two text input fields: 'Email Address' with the value 'mitul@yahoo.com' and 'Password' with masked characters '*****'. Below the fields are three buttons: 'Create Account', 'Login', and 'Cancel'.

- **Create Account Page**

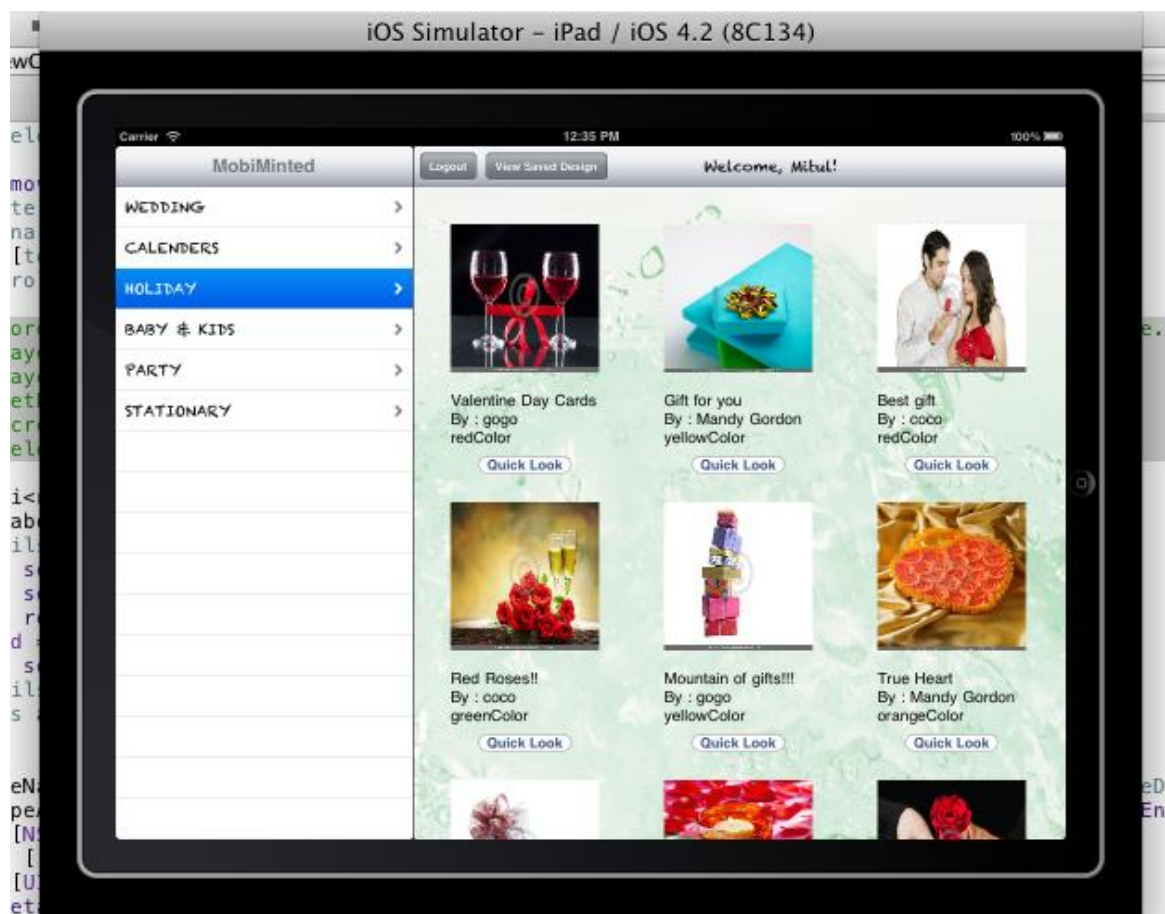
This view is for new user registration. On tapping create account it checks first if all fields are filled or not. Then it checks whether password and confirm password matches or not. Then it checks for username availability and finally if username is available then it creates a new entry in database via SOAP request web service.



The screenshot shows a 'Create Account' dialog box with a white background and a black border. It contains four text input fields: 'User Name' with the value 'harshil', 'email Address' with the value 'harshil@gmail.com', 'Password' with masked characters '*****', and 'Confirm Pass...' with masked characters '*****'. Below the fields are two buttons: 'Create Account' and 'Back'.

- **Template view controller**

This controller provides view of different templates according to chosen subcategory. It provides some unique features like Quick Look in which you can view card but cannot edit. You can also personalize or order sample of that card from that look only. It also displays information like which are colors available and who is the designer.



- **Edit template view controller**

This controller is designed for editing the template you can add label data to your selected template you can choose among diff available paper types and also the quantity. This also provides zoom in feature you can view the template on full screen after editing.



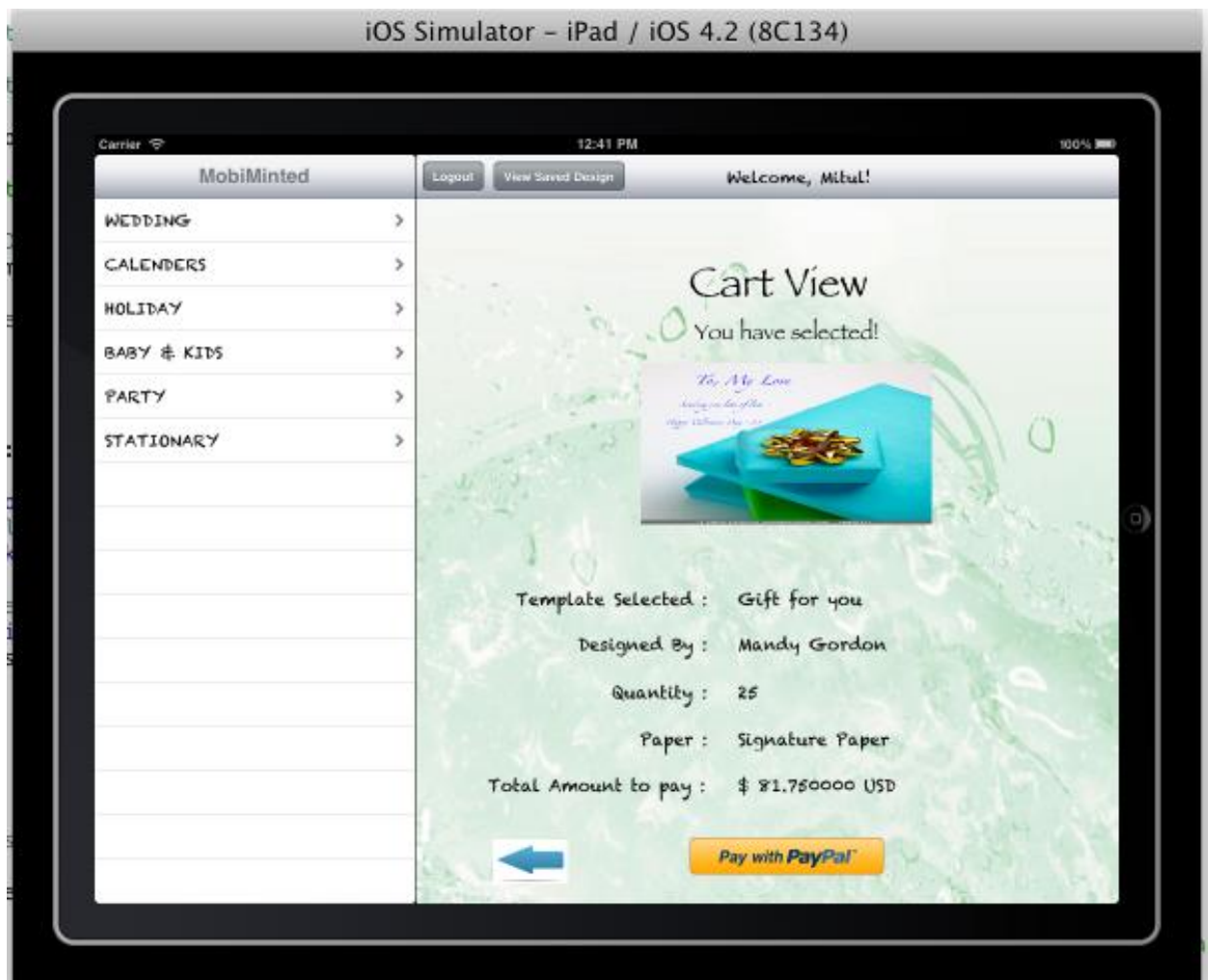
- **XML Parser**

This module is used for parsing the XML data received from server. The data will be received in XML format so we have to retrieve whatever we want. This class

contains three methods one to be called at starting of any tag one for ending and the other is called for data between tag.

- **Cart View Controller**

This class is responsible for giving details of the card user has selected to purchase and to display the total amount to be paid by user. This module also integrates paypal payment options with our application.



- **Temp label**

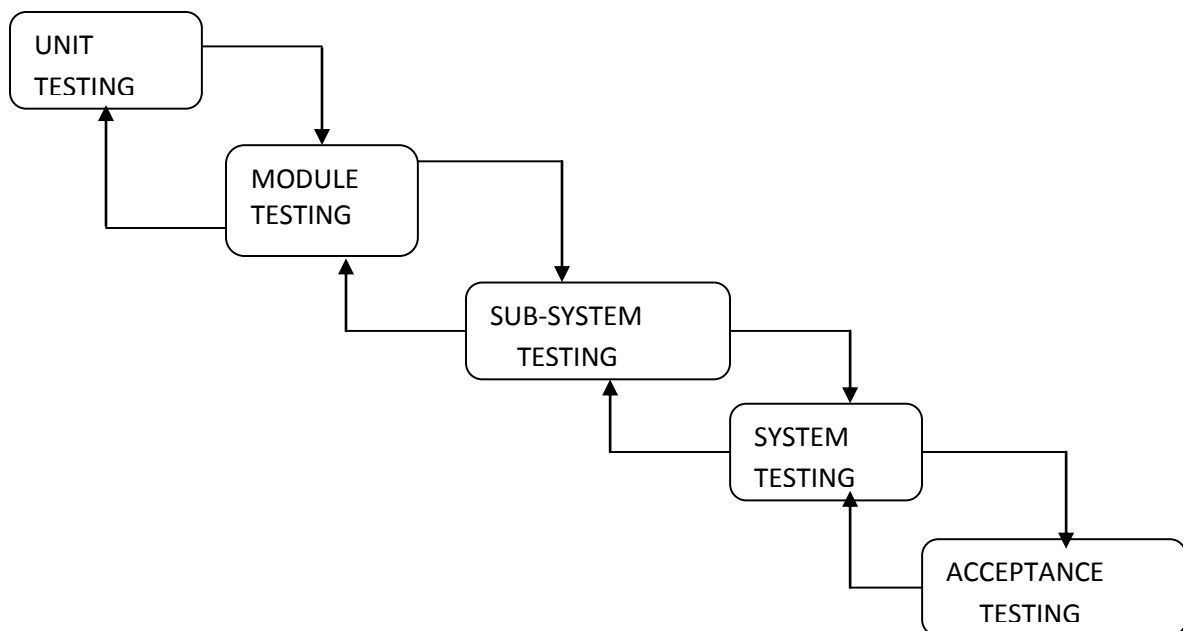
This class is for adding label data to our image of template. This class is inherited from UIImageView. This module mainly does work of drawing text on image.

Testing is the process carried out on software to detect the differences between its actual behavior and the desired behavior as stipulated by the requirements specifications. Testing is advantageous in several ways. Firstly, the defects found help in the process of making the software reliable. Secondly, even if the defects found are not corrected, testing gives an idea as to how reliable the software is. Thirdly, over time, the record of defects found reveals the most common kinds of defects, which can be used for developing appropriate preventive measures such as training, proper design and reviewing

6.1 TESTING PLAN

The testing sub-process includes the following activities in a phase dependent manner:

- Create Test Plans.
- Create Test Specifications.
- Review Test Plans and Test Specifications.
- Conduct tests according to the Test Specifications, and log the defects.
- Fix defects, if any.
- When defects are fixed continue from activity.



6.2 TESTING METHODS

➤ **Black-box and White-box Testing**

In black-box testing a software item is viewed as a black box, without knowledge of its internal structure or behavior. Possible input conditions, based on the specifications (and possible sequences of input conditions), are presented as test cases.

In white-box testing knowledge of internal structure and logic is exploited. Test cases are presented such that possible paths of control flow through the software item are traced. Hence more defects than black-box testing are likely to be found.

The disadvantages are that exhaustive path testing is infeasible and the logic might not conform to specification. Instrumentation techniques can be used to determine the structural system coverage in white box testing. For this purpose tools or compilers that can insert test probes into the programs can be used.

➤ **Code Coverage**

The way to make sure that you have got all the control flow covered is to cover all the paths in the program during the testing (via white-box testing). This implies that both branches are exercised for an 'if' statement, all branches are exercised for a case statement, the loop is taken once or multiple times as well as ignored for a while statement, and all components of complicated logical expressions are exercised. This is called Path Testing. Branch Testing reports whether entire Boolean expression tested in control structures evaluated to both true and false.

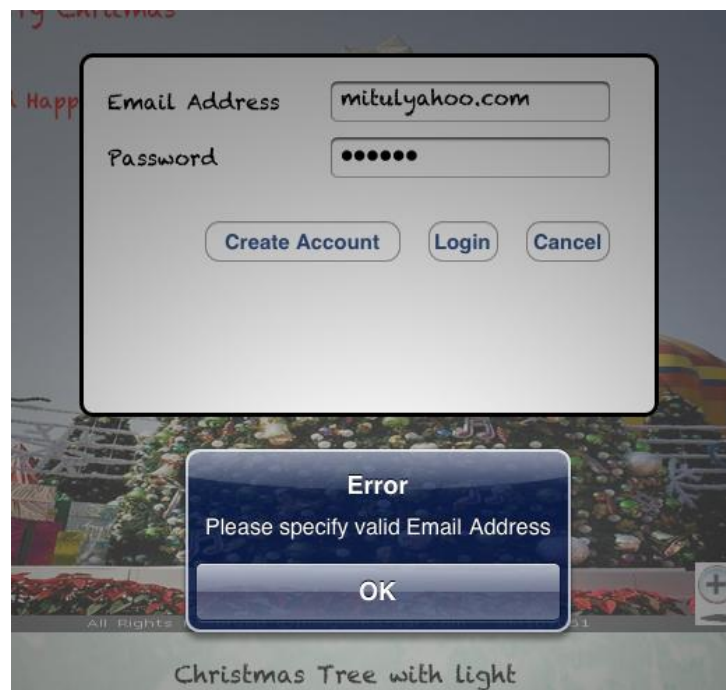
Additionally it includes coverage of switch statement cases, exception handlers and interrupts handlers. Path testing includes branch testing as it

considers all possible combination of individual branch conditions. A simpler version is Statement Testing which determines if each statement in the program has been executed at least once. The coverage via Path Testing includes the coverage via Statement Testing. Since Path Testing is extremely comprehensive it is costly, hence a viable minimum should be measuring Statement Testing coverage.

6.3 TEST CASES

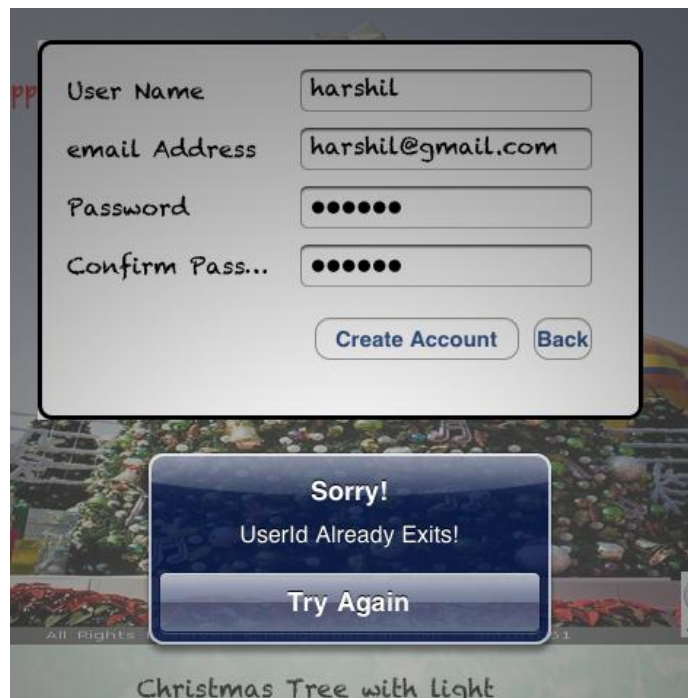
➤ **E-mail address Verification**

This test case is to check whether the user has entered an valid email address or not. Like if user has entered *mitulyahoo.com* then system must generate an alert. This is the response of system when you enter the email address like above.



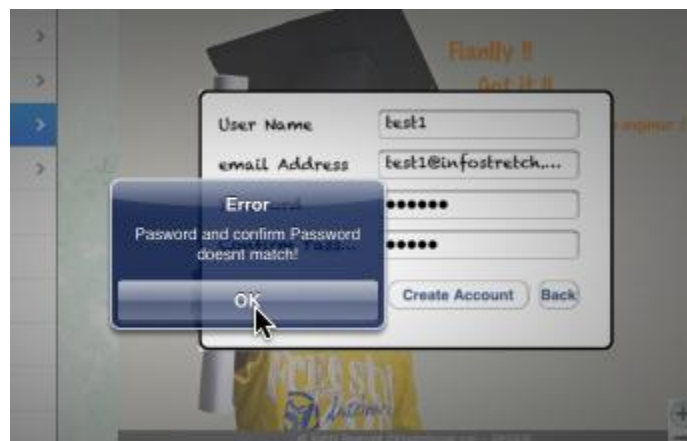
➤ **New user registration**

The email address given by user should not be already registered if so then it should show some message to indicate that.



➤ Password matching

The both field in registration page i.e. password and confirm password should contain same data.



➤ **Authentication**

The email address and password combination should indicate authenticated user. If user enter invalid password then he/she should not be able to login in.



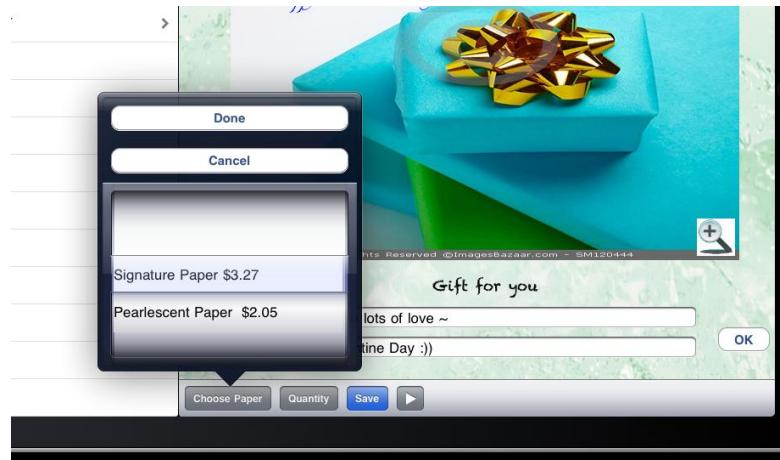
➤ **Login required**

For some of the features of application the user must be logged in like for saving design or to place and order the user must first login himself, if user is not registered then it system should display respective message.



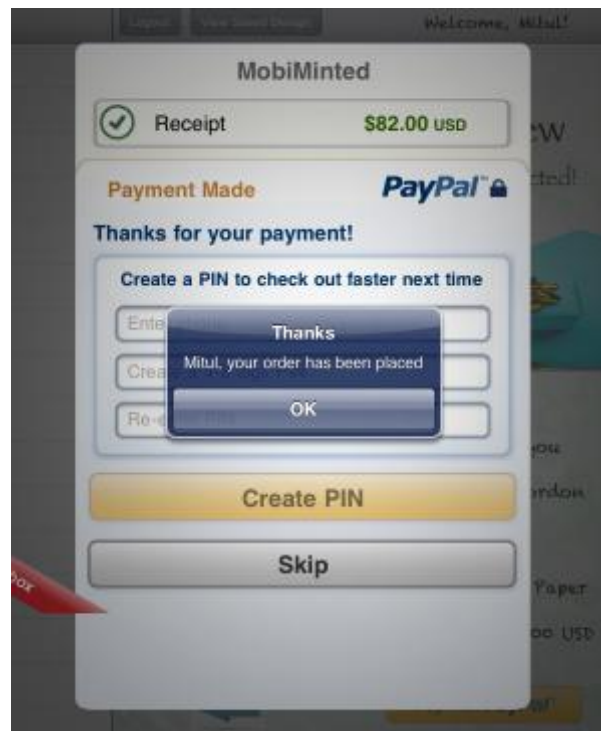
➤ **Choose paper type and quantity**

The paper type and quantity must be chosen before viewing cart as according to type and quantity the total amount to be paid will be calculated.



➤ **Confirmation message**

There must a message to be displayed when user has done payment to confirm that his order has been placed.



➤ **Server crash**

If suppose our server somehow stops working then our application should not make our client wait forever it should display some appropriate message.

We are running a timer for 5 sec if data is not received during this time an alert will be prompted to user to indicate some fault in server.

7.1 CONCLUSION:

The Personalize card module of this project is the core of this project. It will enable user to represent his/her thoughts on card. The objective of the whole project is to give end user ease of designing card for the special occasions of his/her life. If user wants to see demo card, he/she can order by adding card to sample. Since user can save his/her design and view back saved design, user can edit them later on.

If user wants to see all the cards under specific category without editing and personalizing them, user can do so using "Quick Look" facility.

The application has been develop in such a way that it can meet all the specific requirements. If new requirements come, they can be easily implemented.

PayPal service for online payment is integrated in project which is reliable for online payment.

7.2 FUTURE EXTENSIONS:

The application fulfills all the current requirements, but to give an end user more easiness and personalization features some extension may be done which are as follows

- GUI for administrator to add/ remove/ edit templates, category, subcategory.
- Designer himself also should be able to upload his design without admin rights
- Improvement in overall GUI of application.
- Integration of payment gateways other than paypal.

8.1 REFERENCES ARE AS BELOW

- www.developer.apple.com
- www.wikipedia.com
- www.w3schools.com
- www.stackoverflow.com
- www.fourms.sun.com
- Apple's documentation for Objective C and iOS programming.
- Head First iPhone development.
- Software Engineering by Rajib Mall