All-Seeing Eye

1.0

Generated by Doxygen 1.7.4

Thu Aug 11 2011 16:33:20

Contents

Chapter 1

Bug List

Member [cameraView addFrameOverlay:] Overlayed image is fixed size, so this only works on iPad.

Chapter 2

Class Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

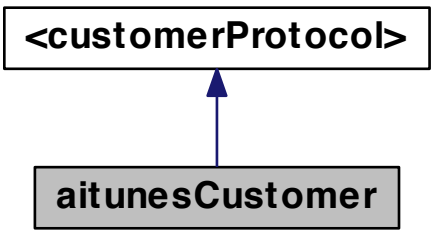
Here is a list of all documented files with brief descriptions:

Chapter 5

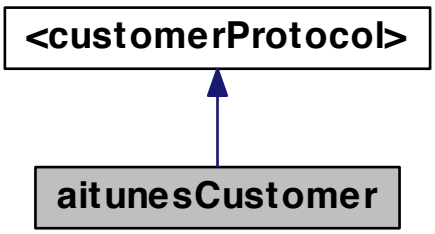
Class Documentation

5.1 aitunesCustomer Class Reference

Inheritance diagram for aitunesCustomer:



Collaboration diagram for aitunesCustomer:



Public Member Functions

- (NSString ∗) - resultFromQuery:inDatabase:

  Return single string from query.
- (int) - customerIdFromDb:withBarcode:

  Get customer ID.
- (NSArray ∗) - customerDefinition

  Return description of customer database.
- (BOOL) - setStringValue:toDb:withBarcode:withFieldType:withTable:withField:

  Set a field value to the DB as a string.
- (NSString ∗) - getStringValueFromDb:withBarcode:withFieldType:withTable:withField:

  Get a field value from the DB as a string.
- (BOOL) - addCustomertoDb:withName:withBarcode:withReferrer:

  Add new customer to database with given name and barcode value.
- (BOOL) - updateLevelOfReferrerWithBarcode:withDb:

  Upgrade customer's level if appropriate. Call for each scan.
- (int) - countOfCustomersInDb:

  Get number of registered customers.
- (NSArray ∗) - allCustomersInDb:

  Get all customers from the database.
- (NSString ∗) - customerFromDb:withBarcode:

  Get customer name.
- (int) - discountFromDb:withBarcode:

  Get customer discount percentage.
- (int) - creditFromDb:withBarcode:

  Get customer monetary credits.

- (BOOL) - clearCreditFromDb:withBarcode:

  Clear customer's credit.
- (int) - referralCountFromDb:withBarcode:

  Get number of customers this customer has referred.
- (int) - countOfOtherBonusesFromDb:withBarcode:

  Get number of bonus rewards customer has.
- (NSString ∗) - otherBonusFromDb:withBarcode:bonusIndex:

  Get the requested bonus reward for a given customer.
- (BOOL) - removeCustomerWithBarcode:fromDb:

  Remove given customer.
- (void) - **_initCustomerDefinition**
- (BOOL) - **updateUserWithBarcode:toDb:withLevel:**
- (BOOL) - **replaceReferralWithBarcode:withReferral:toDb:**

Properties

- NSArray ∗ **_customerDefinition**
- NSString ∗ **lastBarcode**
- NSString ∗ **lastDbFile**

5.1.1 Detailed Description

Definition at line 30 of file aitunesCustomer.h.

5.1.2 Member Function Documentation

5.1.2.1 - (BOOL) addCustomertoDb: dummy(NSString∗) *dbFile* withName:(NSString∗) *name* withBarcode:(NSString∗) *barcode* withReferrer:(NSString∗) *referrer*

Add new customer to database with given name and barcode value.

Parameters

dbFile	Full path to database file
name	Name of new customer
barcode	Barcode of new customer
referrer	Barcode of person who referred customer

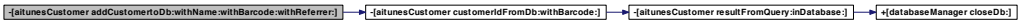
Returns

Yes for success

Reimplemented from <customerProtocol>.

Definition at line 393 of file aitunesCustomer.m.

Here is the call graph for this function:



5.1.2.2 - (NSArray ∗) allCustomersInDb: dummy(NSString∗) *dbFile*

Get all customers from the database.

Parameters

dbFile	Database to query

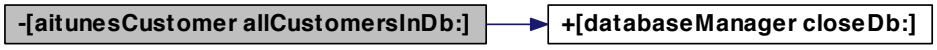
Returns

Dictionary of all customers, with name as key and barcode as value

Reimplemented from <customerProtocol>.

Definition at line 611 of file aitunesCustomer.m.

Here is the call graph for this function:



5.1.2.3 - (BOOL) clearCreditFromDb: dummy(NSString∗) *dbFile* withBarcode:(NSString∗) *barcode*

Clear customer's credit.

Parameters

dbFile	Database to search
barcode	Barcode number to match

Returns

Clear's customer's monetary credit to zero

Reimplemented from <customerProtocol>.

Definition at line 726 of file aitunesCustomer.m.

Here is the call graph for this function:

5.1.2.4 - (int) countOfCustomersInDb: dummy(NSString∗) *dbFile*

Get number of registered customers.

Parameters

dbFile	Database to search

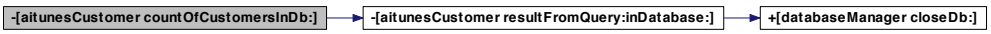
Returns

Number of customers

Reimplemented from <customerProtocol>.

Definition at line 598 of file aitunesCustomer.m.

Here is the call graph for this function:



5.1.2.5 - (int) countOfOtherBonusesFromDb: dummy(NSString∗) *dbFile* withBarcode:(NSString∗) *barcode*

Get number of bonus rewards customer has.

Parameters

dbFile	Database to search
barcode	Barcode number to match

Returns

Number of bonus rewards customer has

Reimplemented from <customerProtocol>.

Definition at line 743 of file aitunesCustomer.m.

5.1.2.6 - (int) creditFromDb: dummy(NSString∗) *dbFile* withBarcode:(NSString∗) *barcode*

Get customer monetary credits.

Parameters

dbFile	Database to search
barcode	Barcode number to match

Returns

Customer's monetary credits

Reimplemented from <customerProtocol>.

Definition at line 709 of file aitunesCustomer.m.

Here is the call graph for this function:

5.1.2.7 - (NSArray ∗) customerDefinition

Return description of customer database.

See customerProtocol.h for implementation details. Returns copy of private variable.

Returns

Data structure representing customer database

Reimplemented from <customerProtocol>.

Definition at line 132 of file aitunesCustomer.m.

5.1.2.8 - (NSString ∗) customerFromDb: dummy(NSString∗) *dbFile* withBarcode:(NSString∗) *barcode*

Get customer name.

Parameters

dbFile	Database to search
barcode	Barcode number to match

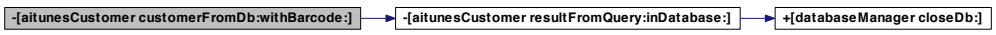
Returns

Customer name

Reimplemented from <customerProtocol>.

Definition at line 648 of file aitunesCustomer.m.

Here is the call graph for this function:



5.1.2.9 - (int) customerIdFromDb: dummy(NSString*) *dbFile* withBarcode:(NSString*) *barcode*

Get customer ID.

Parameters

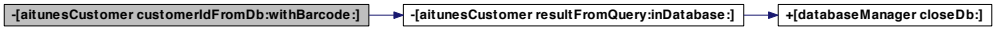
dbFile	Database file to search
barcode	Barcode number of customer in question

Returns

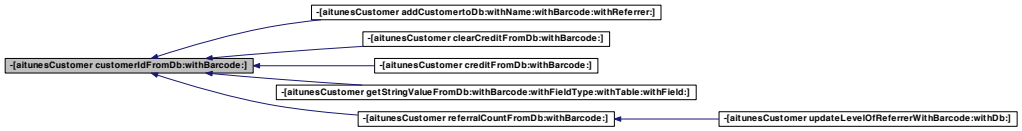
Customer ID, or -1 on error

Definition at line 584 of file aitunesCustomer.m.

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.2.10 - (int) discountFromDb: dummy(NSString*) *dbFile* withBarcode:(NSString*) *barcode*

Get customer discount percentage.

Parameters

dbFile	Database to search
barcode	Barcode number to match

Returns

Customer discount percentage

Reimplemented from <customerProtocol>.

Definition at line 684 of file aitunesCustomer.m.

5.1.2.11 - (NSString ∗) getStringValueFromDb: dummy(NSString∗) *dbFile* withBarcode:(NSString∗) *barcode* withFieldType:(NSString∗) *type* withTable:(NSString∗) *table* withField:(NSString∗) *field*

Get a field value from the DB as a string.

Generic, abstract interface to fetch the value of any field from the customer database using data that the UI can get by other customerProtocol queries. Specifically, 'type','table', and 'field' should be a valid combination from a dictionary in customerDefinition. Customer to query is determined by the barcode.

Parameters

dbFile	Database file to search in
barcode	Barcode of customer to get information on
type	Type of field being queried (determines string formatting)
table	Table in database to query
field	Field in database to query

Returns

String value of requested field.

Reimplemented from <customerProtocol>.

Definition at line 294 of file aitunesCustomer.m.

Here is the call graph for this function:

5.1.2.12 - (NSString ∗) otherBonusFromDb: dummy(NSString∗) *dbFile* withBarcode:(NSString∗) *barcode* bonusIndex:(int) *idx*

Get the requested bonus reward for a given customer.

In addition to percent discount and monetary credits, a customer can have any number of miscellaneous earned bonus rewards. Since these aren't specifically understood by the system, they are simply stored as strings. This function returns a string describing a specific reward.

Parameters

dbFile	Database to search
barcode	Barcode number to match

Returns

A customer's specific bonus reward (as string)

Reimplemented from <customerProtocol>.

Definition at line 760 of file aitunesCustomer.m.

5.1.2.13 - (int) referralCountFromDb: dummy(NSString*) *dbFile* withBarcode:(NSString*) *barcode*

Get number of customers this customer has referred.

Parameters

dbFile	Database to search
barcode	Barcode of customer

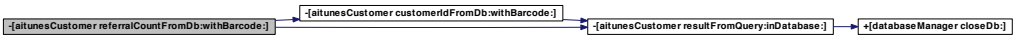
Returns

Customer's referral count

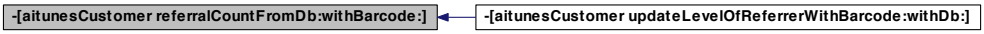
Reimplemented from <customerProtocol>.

Definition at line 566 of file aitunesCustomer.m.

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.2.14 - (BOOL) removeCustomerWithBarcode: dummy(NSString∗) *barcode* fromDb:(NSString∗) *dbFile*

Remove given customer.

Parameters

barcode	Barcode of customer to remove
dbFile	Database to search

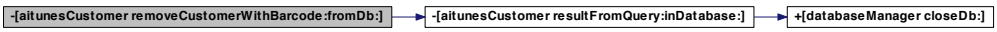
Returns

Whether customer was successfully removed

Reimplemented from <customerProtocol>.

Definition at line 772 of file aitunesCustomer.m.

Here is the call graph for this function:



5.1.2.15 - (NSString ∗) resultFromQuery: dummy(NSString∗) *query* inDatabase:(NSString∗) *dbFile*

Return single string from query.

Returns result of database query. The query should return a single row, with the desired result in the first column. This method returns a single value, not an entire row.

Parameters

query	Query to execute
dbFile	Database file to query

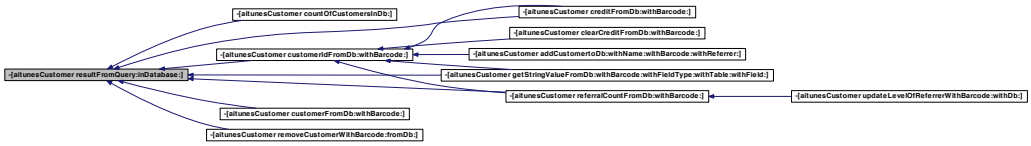
Returns

Result of query, or nil on error

Definition at line 100 of file aitunesCustomer.m.

Here is the call graph for this function:

Here is the caller graph for this function:



5.1.2.16 - (BOOL) setStringValue: dummy(NSString∗) *text* toDb:(NSString∗) *dbFile* withBarcode:(NSString∗) *barcode* withFieldType:(NSString∗) *type* withTable:(NSString∗) *table* withField:(NSString∗) *field*

Set a field value to the DB as a string.

Generic, abstract interface to set the value of any field from the customer database using data that the UI can get by other customerProtocol queries. Specifically, 'type','table', and 'field' should be a valid combination from a dictionary in customerDefinition. Customer to query is determined by the barcode.

Parameters

text	Text to set field to
dbFile	Database file to write to
barcode	Barcode of customer to modify
type	Type of field being queried (determines string formatting)
table	Table in database to query
field	Field in database to query

Returns

String value of requested field.

Reimplemented from <customerProtocol>.

Definition at line 340 of file aitunesCustomer.m.

5.1.2.17 - (BOOL) updateLevelOfReferrerWithBarcode: dummy(NSString∗) *barcode* withDb:(NSString∗) *dbFile*

Upgrade customer's level if appropriate. Call for each scan.

Parameters

barcode	Barcode of customer

Returns

Yes if no errors, No if error encountered.

Reimplemented from <customerProtocol>.

Definition at line 443 of file aitunesCustomer.m.

Here is the call graph for this function:

The documentation for this class was generated from the following files:

- aitunesCustomer.h
- aitunesCustomer.m

5.2 cameraView Class Reference

Cropped view of rear camera video stream.

Public Member Functions

- (id) - initWithFrame:

  Initialize view with a rectangle.
- (void) - initCapture

  Start video capture for this view.
- (CGImageRef) - cropImage:

  Crop a video frame.
- (UIImage ∗) - addFrameOverlay:

  Adds a 'target' overlay to image frame.
- (void) - captureOutput:didOutputSampleBuffer:fromConnection:

  Delegate to receive, display, and scan incoming video frames.

Properties

- UIImageView ∗ imageView
- AVCaptureSession ∗ captureSession

5.2.1 Detailed Description

Cropped view of rear camera video stream.

This UIView subclass presents a rectangular, live video view from the default camera. The frames captured from the camera are set as the background image of the view.

This class also implements the delegate of a video capture device, and the method that receieves each video frame passes them off to the

Each frame is cropped to (SCREEN_WIDTH)x(SCREEN_HEIGHT/4), but note that it will happily display outside of its bounds box, so this class should be initialized with the correct size region given to initWithFrame:.

Definition at line 37 of file cameraView.h.

5.2.2 Member Function Documentation

5.2.2.1 - (UIImage ∗) addFrameOverlay: dummy(UIImage∗) *baseImg*

Adds a 'target' overlay to image frame.

Given a cropped video frame, overlays a translucent 'target' image over it to give the viewer an indication of where the barcode should be.

Bug

Overlayed image is fixed size, so this only works on iPad.

Parameters

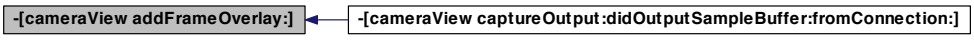
baseImg	Cropped image to add overlay to

Returns

New image with overlay

Definition at line 166 of file cameraView.m.

Here is the caller graph for this function:



5.2.2.2 - (void) captureOutput: dummy(AVCaptureOutput ∗) *captureOutput* didOutputSampleBuffer:(CMSampleBufferRef) *sampleBuffer* fromConnection:(AVCaptureConnection ∗) *connection*

Delegate to receive, display, and scan incoming video frames.

Called whenever the camera has a new frame captured, this method crops, overlays, rotates, and resizes the frame before displaying it. It also passes the frame to the global barcode scanner to check if any barcodes are readable.

Parameters

captureOutput	Output device that generated the frame
sampleBuffer	Raw data returned from the camera
connection	Unused.

Definition at line 194 of file cameraView.m.

Here is the call graph for this function:

5.2.2.3 - (CGImageRef) cropImage: dummy(CGImageRef) *img*

Crop a video frame.

Crop height to 1/4th of screen height.

NOTE: Frame is rotated when received from camera, so this function actually sets the "width" to SCREEN_HEIGHT/4, since the width of the unrotated image will become the height of the rotated one.

Parameters

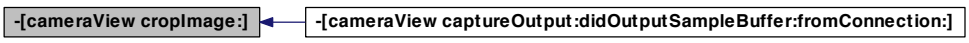
img	Image to crop (frame from video capture).

Returns

Cropped image

Definition at line 147 of file cameraView.m.

Here is the caller graph for this function:



5.2.2.4 - (void) initCapture

Start video capture for this view.

Connects to video input device, sets local delegate method as video output handler, configures frame queue, autofocus, framerate, etc,

Parameters

\return	

Definition at line 86 of file cameraView.m.

5.2.2.5 - (id) initWithFrame: dummy(CGRect) *aRect*

Initialize view with a rectangle.

Creates an image with the same size as the given bound rectangle, and sets that image as the only subview of cameraView. The image will be changed to be the most recent frame from the video when the video is running.

NOTE: This class has ∗only∗ been tested with the input rectangle: **(SCREEN_WIDTH, SCREEN_HEIGHT/4)**

Other sizes may or may not work without code changes. This does work correctly on both iPad and iPhone, though.

Parameters

aRect	Bounding rectangle for this view.

Returns

Initialized instance of class

Definition at line 65 of file cameraView.m.

5.2.3 Property Documentation

5.2.3.1 - (AVCaptureSession ∗) captureSession [read, write, retain]

Video capture session used to interact with camera.

Definition at line 39 of file cameraView.h.

5.2.3.2 - (UIImageView ∗) imageView [read, write, retain]

Image view filled with video frame and displayed in cameraView.

Definition at line 38 of file cameraView.h.

The documentation for this class was generated from the following files:

- cameraView.h
- cameraView.m

5.3 codeScanner Class Reference

Handles scanning images for barcodes.

Public Member Functions

- (void) - **simulatorDebug**
- (BOOL) - **scanImage:**

Properties

- ZBarImageScanner * scanner

  ZBar barcode scanner instance.
- NSString * lastCode

  String of the last barcode scan result.

5.3.1 Detailed Description

Handles scanning images for barcodes.

This class keeps an instance of the ZBar barcode scanner. It receives images from the cameraView and scans them for barcodes. If a barcode is detected, the decoded result is stored in an instance variable for decoding.

Definition at line 28 of file codeScanner.h.

The documentation for this class was generated from the following files:

- codeScanner.h
- codeScanner.m

5.4 customerInfoView Class Reference

Public Member Functions

- (id) - initWithFrame:

  Initialize view to given size.
- (void) - newScanHandler:

  Handles new successful scan events.
- (void) - redrawScreen

Tells view to redraw itself.

- (void) - drawRect:

  Called when view needs to be redrawn.

- (void) - drawCenteredText:y:

  Draw centered text at the given y-coordinate.

- (void) - drawLeftJustifiedText:y:

  Draw left-justified text at the given y-coordinate.

- (void) - drawCenteredImage:y:

  Draws an image on the view, centered on screen, at given y-coordinate.

- (void) - drawLeftJustifiedImage:y:

  Draws an image on the view, left-justified, at given y-coordinate.

- (UIImage ∗) - imageFromText:withMaxFontSize:

  Creates an image containing the given text.

- (void) - **displayInvalidScanNotification**
- (void) - **scanTimerCallback:**
- (void) - **scheduleScanTimeout**
- (void) - **enableRedeemButton**
- (void) - **disableRedeemButton**

5.4.1 Detailed Description

Definition at line 26 of file customerInfoView.h.

5.4.2 Member Function Documentation

5.4.2.1 - (void) drawCenteredImage: dummy(UIImage∗) *img* y:(int) *y*

Draws an image on the view, centered on screen, at given y-coordinate.

Parameters

img	Image to draw in view
y	y coordinate to draw image

Definition at line 436 of file customerInfoView.m.

Here is the caller graph for this function:

5.4.2.2 - (void) drawCenteredText: dummy(NSString∗) *str* y:(int) *y*

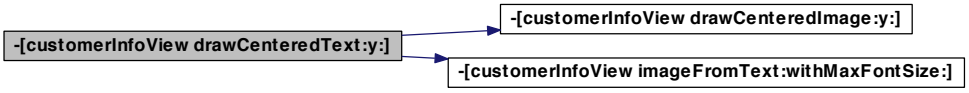
Draw centered text at the given y-coordinate.

Parameters

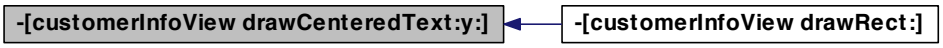
str	Text to draw on screen
y	y coordinate to draw text

Definition at line 412 of file customerInfoView.m.

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.2.3 - (void) drawLeftJustifiedImage: dummy(UIImage∗) *img* y:(int) *y*

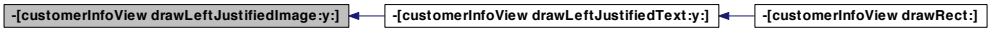
Draws an image on the view, left-justified, at given y-coordinate.

Parameters

img	Image to draw in view
y	y coordinate to draw image

Definition at line 450 of file customerInfoView.m.

Here is the caller graph for this function:



5.4.2.4 - (void) drawLeftJustifiedText: dummy(NSString∗) *str* y:(int) *y*

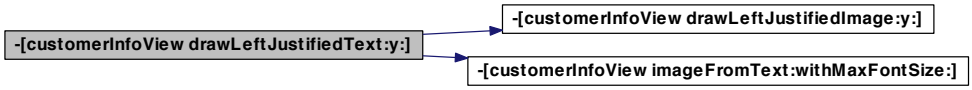
Draw left-justified text at the given y-coordinate.

Parameters

str	Text to draw on screen
y	y coordinate to draw text

Definition at line 424 of file customerInfoView.m.

Here is the call graph for this function:



Here is the caller graph for this function:

5.4.2.5 - (void) drawRect: dummy(CGRect) *rect*

Called when view needs to be redrawn.

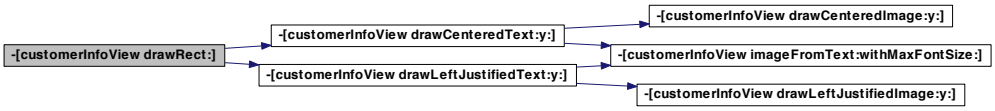
Main GUI thread calls this when view needs to be redrawn. This controls what text is displayed in the view, and where. Should be called each time a new barcode is successfully scanned. That happens automatically via the notification system.

Parameters

rect	Region to redraw. Ignored, always full view.

Definition at line 341 of file customerInfoView.m.

Here is the call graph for this function:



5.4.2.6 - (UIImage ∗) imageFromText: dummy(NSString ∗) *text* withMaxFontSize:(float) *maxFont*

Creates an image containing the given text.

Given a string, this creates an image containing the text with a glowing shadow, ready to be drawn on the view. Draws with a large font, but automatically shrinks font to text will fit on the screen, down to a small size that is still readable on iPhone and iPad. If the text won't fit with the minimum font size, it is truncated with ellipses.

Parameters

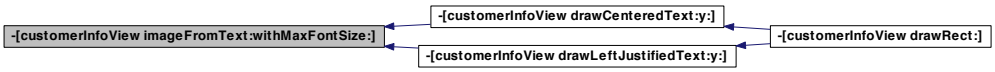
text	String to convert to image

Returns

Image containing formatted text

Definition at line 469 of file customerInfoView.m.

Here is the caller graph for this function:



5.4.2.7 - (id) initWithFrame: dummy(CGRect) *aRect*

Initialize view to given size.

Creates view with given bounds. Background color set to dark gray, and registers for ASE_BarcodeScanned notification events. Sets default text values to display.

Parameters

aRect	Size of view

Returns

Initialized instance of view

Definition at line 83 of file customerInfoView.m.

5.4.2.8 - (void) newScanHandler: dummy(NSNotification *) *notif*

Handles new successful scan events.

Called when a barcode is successfully scanned (by notification system), this just sets redrawScreen to be called on the main thread.

Parameters

notif	Notification that caused this to run

Definition at line 191 of file customerInfoView.m.

The documentation for this class was generated from the following files:

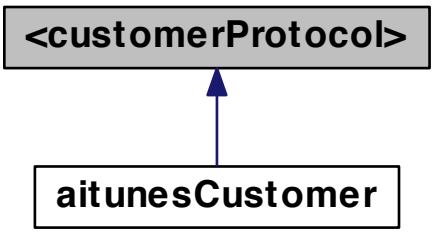
- customerInfoView.h
- customerInfoView.m

5.5 <customerProtocol> Protocol Reference

Protocol for requesting customer data.

`#import <customerProtocol.h>`

Inheritance diagram for <customerProtocol>:



Public Member Functions

- (NSArray ∗) - customerDefinition

  Return description of customer database.
- (NSString ∗) - getStringValueFromDb:withBarcode:withFieldType:withTable:withField:

  Get a field value from the DB as a string.
- (BOOL) - setStringValue:toDb:withBarcode:withFieldType:withTable:withField:

  Set a field value to the DB as a string.
- (BOOL) - addCustomertoDb:withName:withBarcode:withReferrer:

  Add new customer to database with given name and barcode value.
- (BOOL) - updateLevelOfReferrerWithBarcode:withDb:

  Upgrade customer's level if appropriate. Call for each scan.
- (int) - countOfCustomersInDb:

  Get number of registered customers.
- (NSArray ∗) - allCustomersInDb:

  Get all customers from the database.
- (NSString ∗) - customerFromDb:withBarcode:

  Get customer name.
- (int) - levelFromDb:withBarcode:

  Get customer rewards level.
- (int) - discountFromDb:withBarcode:

  Get customer discount percentage.
- (int) - creditFromDb:withBarcode:

  Get customer monetary credits.
- (BOOL) - clearCreditFromDb:withBarcode:

  Clear customer's credit.
- (int) - referralCountFromDb:withBarcode:

  Get number of customers this customer has referred.
- (int) - countOfOtherBonusesFromDb:withBarcode:

  Get number of bonus rewards customer has.
- (NSString ∗) - otherBonusFromDb:withBarcode:bonusIndex:

  Get the requested bonus reward for a given customer.
- (BOOL) - removeCustomerWithBarcode:fromDb:

  Remove given customer.

5.5.1 Detailed Description

Protocol for requesting customer data.

This file defines a protocol to be implemented by classes that provide data about a customer via database lookups. A class that directly accesses the database should implement this protocol, and expect to be called by a view that handles displaying customer data.

This is split out as a protocol to support the idea of deploying to multiple venues with minimal code change. The database specifics and reward levels can change without changing the frontend view. It does, however, limit the reward system to having levels, percent discounts, and monetary credits.

Certain information is mandatory (name, level, etc), but this protocol also provides mechanisms for adding additional information to each customer account, and generalizes it such that a UI implementing this protocol should be able to support fairly large customer descriptions. The generic system only supports 1-1 mappings, however.

Definition at line 47 of file customerProtocol.h.

5.5.2 Member Function Documentation

5.5.2.1 - (BOOL) addCustomertoDb: dummy(NSString *) *dbFile* withName:(NSString *) *name* withBarcode:(NSString *) *barcode* withReferrer:(NSString *) *referrer*

Add new customer to database with given name and barcode value.

Parameters

dbFile	Full path to database file
name	Name of new customer
barcode	Barcode of new customer

Returns

Yes for success

Reimplemented in aitunesCustomer.

5.5.2.2 - (NSArray*) allCustomersInDb: dummy(NSString *) *dbFile*

Get all customers from the database.

Parameters

dbFile	Database to query

Returns

Dictionary of all customers, with name as key and barcode as value

Reimplemented in aitunesCustomer.

5.5.2.3 - (BOOL) clearCreditFromDb: dummy(NSString *) *dbFile* withBarcode:(NSString *) *barcode*

Clear customer's credit.

Parameters

dbFile	Database to search
barcode	Barcode number to match

Returns

Clear's customer's monetary credit to zero

Reimplemented in aitunesCustomer.

5.5.2.4 - (int) countOfCustomersInDb: dummy(NSString *) *dbFile*

Get number of registered customers.

Parameters

dbFile	Database to search

Returns

Number of customers

Reimplemented in aitunesCustomer.

5.5.2.5 - (int) countOfOtherBonusesFromDb: dummy(NSString *) *dbFile* withBarcode:(NSString *) *barcode*

Get number of bonus rewards customer has.

Parameters

dbFile	Database to search
barcode	Barcode number to match

Returns

Number of bonus rewards customer has

Reimplemented in aitunesCustomer.

5.5.2.6 - (int) creditFromDb: dummy(NSString *) *dbFile* withBarcode:(NSString *) *barcode*

Get customer monetary credits.

Parameters

dbFile	Database to search
barcode	Barcode number to match

Returns

Customer's monetary credits

Reimplemented in aitunesCustomer.

5.5.2.7 - (NSArray *) customerDefinition

Return description of customer database.

Returns a deeply-nested data structure that defines the information stored in the customer database. This data structure is used as the basis for creating and editing customer entries, and describes everything the application needs to display, get, and set customer info.

The format of the data structure is as follows:

The top-level returned object is an array of Section Names and Sections. Sections define which fields should be grouped together when displayed. For example, a "contact information" section might group the phone and street address.

The top-level array alternates section name (NSString*), section (NSArray*), section name, section... two entries per section.

The section array contains a variable number of Row dictionaries. Rows represent a single piece of information about a customer, and map to a field in the database.

The row dictionary keys can be:

- cellName -- Name to display as cell label on UI
- cellType -- Type of data represented
- dbTable -- Name of table in db that stores this field
- dbField -- Name of corresponding field in db's table
- required -- Whether this field is required

Returns

Data structure representing customer database

Reimplemented in aitunesCustomer.

5.5.2.8 - (NSString*) customerFromDb: dummy(NSString *) *dbFile* withBarcode:(NSString *) *barcode*

Get customer name.

Parameters

dbFile	Database to search
barcode	Barcode number to match

Returns

Customer name

Reimplemented in aitunesCustomer.

5.5.2.9 - (int) discountFromDb: dummy(NSString ∗) *dbFile* withBarcode:(NSString ∗) *barcode*

Get customer discount percentage.

Parameters

dbFile	Database to search
barcode	Barcode number to match

Returns

Customer discount percentage

Reimplemented in aitunesCustomer.

5.5.2.10 - (NSString∗) getStringValueFromDb: dummy(NSString ∗) *dbFile* withBarcode:(NSString ∗) *barcode* withFieldType:(NSString ∗) *type* withTable:(NSString ∗) *table* withField:(NSString ∗) *field*

Get a field value from the DB as a string.

Generic, abstract interface to fetch the value of any field from the customer database using data that the UI can get by other customerProtocol queries. Specifically, 'type','table', and 'field' should be a valid combination from a dictionary in customerDefinition. Customer to query is determined by the barcode.

Parameters

dbFile	Database file to search in
barcode	Barcode of customer to get information on
type	Type of field being queried (determines string formatting)
table	Table in database to query
field	Field in database to query

Returns

String value of requested field.

Reimplemented in aitunesCustomer.

5.5.2.11 - (int) levelFromDb: dummy(NSString ∗) *dbFile* withBarcode:(NSString ∗) *barcode*

Get customer rewards level.

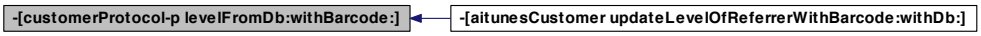
Parameters

dbFile	Database to search
barcode	Barcode number to match

Returns

Customer rewards level

Here is the caller graph for this function:



5.5.2.12 - (NSString∗) otherBonusFromDb: dummy(NSString ∗) *dbFile* withBarcode:(NSString ∗) *barcode* bonusIndex:(int) *idx*

Get the requested bonus reward for a given customer.

In addition to percent discount and monetary credits, a customer can have any number of miscellaneous earned bonus rewards. Since these aren't specifically understood by the system, they are simply stored as strings. This function returns a string describing a specific reward.

Parameters

dbFile	Database to search
barcode	Barcode number to match

Returns

A customer's specific bonus reward (as string)

Reimplemented in aitunesCustomer.

5.5.2.13 - (int) referralCountFromDb: dummy(NSString ∗) *dbFile* withBarcode:(NSString ∗) *barcode*

Get number of customers this customer has referred.

Parameters

dbFile	Database to search
barcode	Barcode of customer

Returns

Customer's referral count

Reimplemented in aitunesCustomer.

5.5.2.14 - (BOOL) removeCustomerWithBarcode: dummy(NSString *) *barcode* fromDb:(NSString *) *dbFile*

Remove given customer.

Parameters

barcode	Barcode of customer to remove
dbFile	Database to search

Returns

Whether customer was successfully removed

Reimplemented in aitunesCustomer.

5.5.2.15 - (BOOL) setStringValue: dummy(NSString *) *text* toDb:(NSString *) *dbFile* withBarcode:(NSString *) *barcode* withFieldType:(NSString *) *type* withTable:(NSString *) *table* withField:(NSString *) *field*

Set a field value to the DB as a string.

Generic, abstract interface to set the value of any field from the customer database using data that the UI can get by other customerProtocol queries. Specifically, 'type','table', and 'field' should be a valid combination from a dictionary in customerDefinition. Customer to query is determined by the barcode.

Parameters

text	Text to set field to
dbFile	Database file to write to
barcode	Barcode of customer to modify
type	Type of field being queried (determines string formatting)
table	Table in database to query
field	Field in database to query

Returns

String value of requested field.

Reimplemented in aitunesCustomer.

5.5.2.16 - (BOOL) updateLevelOfReferrerWithBarcode: dummy(NSString *) *barcode* withDb:(NSString *) *dbFile*

Upgrade customer's level if appropriate. Call for each scan.

Parameters

barcode	Barcode of customer

Returns

Yes if no errors, No if error encountered.

Reimplemented in aitunesCustomer.

The documentation for this protocol was generated from the following file:

- customerProtocol.h

5.6 databaseManager Class Reference

Public Member Functions

- (id) - initWithFile:

  Create instance, and create new database file if needed.
- (BOOL) - reloadWithNewDatabaseFile:

  Copies over existing database with a new database file.
- (BOOL) - logString:

  Write a string to the open log file and flush to disk.
- (NSString ∗) - **pathFromFile:**
- (BOOL) - **copyDatabaseToDocuments**
- (void) - **generateLogFileNameAndOpen**
- (void) - **closeGlobalDB**

Static Public Member Functions

- (BOOL) + openDbFile:usingDbPointer:

  Opens a sqlite database file.
- (void) + closeDb:

  Closes given database connection if it's open.

Properties

- NSString ∗ databasePath

  Full path and filename of database.
- NSString ∗ logFile

  Full path and filename of log file.
- NSString ∗ logPrefix

  String prefix for log files.

5.6.1 Detailed Description

Definition at line 28 of file databaseManager.h.

5.6.2 Member Function Documentation

5.6.2.1 + (void) closeDb: dummy(sqlite3∗∗) *db*

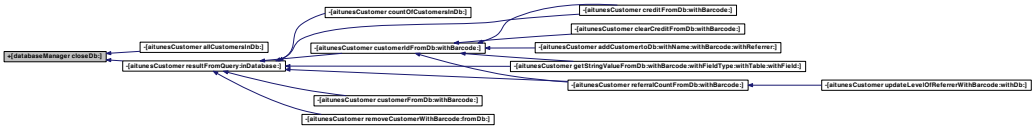
Closes given database connection if it's open.

Parameters

db	Pointer to database handle

Definition at line 251 of file databaseManager.m.

Here is the caller graph for this function:



5.6.2.2 - (id) initWithFile: dummy(NSString∗) *filename*

Create instance, and create new database file if needed.

Given a filename, this initializes a database manager that uses that file in the user's local Documents directory. If a database does not already exist in that directory, a new db is copied there from the application bundle.

Parameters

filename	Filename of customer database

Returns

Initialized instance

Definition at line 74 of file databaseManager.m.

5.6.2.3 - (BOOL) logString: dummy(NSString∗) *str*

Write a string to the open log file and flush to disk.

Parameters

str	String to write to log file

Returns

Yes on success, no if log file is not open

Definition at line 147 of file databaseManager.m.

5.6.2.4 + (BOOL) openDbFile: dummy(NSString∗) *file* usingDbPointer:(sqlite3∗∗) *db*

Opens a sqlite database file.

Parameters

file	Full path to database file
db	Handle to make reference to open database (output parameter)

Returns

Whether open succeeded

Definition at line 241 of file databaseManager.m.

5.6.2.5 - (BOOL) reloadWithNewDatabaseFile: dummy(NSURL∗) *url*

Copies over existing database with a new database file.

This copies a new database file, specified in the 'url' argument, over the existing database. It is an overwrite operation, so the old db is lost forever.

This method is expected to be called when an external application, such as an e-mail client, delegates All-Seeing Eye to open a database.

Parameters

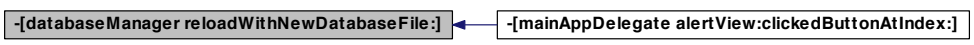
url	Full path to new database file

Returns

Whether overwrite was successful

Definition at line 172 of file databaseManager.m.

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- databaseManager.h
- databaseManager.m

5.7 dateInputVC Class Reference

Public Member Functions

- (void) - **addCancelButton**
- (void) - **addDoneButton**

Properties

- UIDatePicker * **pickerView**
- id delegate

  Delegate implementing textInputVCProtocol.
- id userData

  Identifier data passed back to delegate after an event occurs.

5.7.1 Detailed Description

Definition at line 27 of file dateInputVC.h.

The documentation for this class was generated from the following file:

- dateInputVC.h

5.8 <dateInputVCProtocol> Protocol Reference

Public Member Functions

- (void) - dateInputView:withUserData:updatedText:

  Protocol describing how dateInputVC informs delegate of text change.

5.8.1 Detailed Description

Definition at line 43 of file dateInputVC.h.

The documentation for this protocol was generated from the following file:

- dateInputVC.h

5.9 dropboxSync Class Reference

Public Member Functions

- (BOOL) - openDropboxSession

  Call to connect to Dropbox service.
- (void) - getLockAndWriteDatabase:

  Request lock and write database to Dropbox (blocking)
- (void) - writeDatabaseToDropbox:

  Request that Database be written to Dropbox This is asynchronous. Returns before database is uploaded.
- (BOOL) - tryToObtainDropboxLock

  Attempts to create folder on Dropbox to obtain write lock Asynchronous, returns before lock obtained.
- (void) - releaseDropboxLock

  Attempts to delete folder on Dropbox to release write lock Asynchronous, returns before lock released.
- (void) - **openDropboxLoginWindow**
- (DBRestClient ∗) - **initRestClient**
- (void) - **clearDropboxCredentials**
- (BOOL) - **saveDropboxCredentials**
- (void) - **readDatabaseFromDropbox**
- (void) - **periodicDatabaseDownloadThread**

Properties

- DBRestClient ∗ **restClient**
- BOOL **hasWriteLock**
- BOOL **hasLockPermission**
- BOOL **uploadInProgress**

5.9.1 Detailed Description

Definition at line 27 of file dropboxSync.h.

5.9.2 Member Function Documentation

5.9.2.1 - (void) getLockAndWriteDatabase: dummy(NSString∗) *localPath*

Request lock and write database to Dropbox (blocking)

This is a BLOCKING call to get the lock, upload the database, and unlock.

Parameters

localPath	Path to local file to upload

Definition at line 340 of file dropboxSync.m.

5.9.2.2 - (BOOL) openDropboxSession

Call to connect to Dropbox service.

Creates Dropbox session. Gives Dropbox login screen if no credentials are stored. If credentials are stored locally, prompts user for whether he wants to use them.

Returns

Yes if link has been established.

Definition at line 113 of file dropboxSync.m.

5.9.2.3 - (BOOL) tryToObtainDropboxLock

Attempts to create folder on Dropbox to obtain write lock Asynchronous, returns before lock obtained.

Returns

No on error, yes if request scheduled.

Definition at line 419 of file dropboxSync.m.

5.9.2.4 - (void) writeDatabaseToDropbox: dummy(NSString∗) *localPath*

Request that Database be written to Dropbox This is asynchronous. Returns before database is uploaded.

Parameters

localPath	Local file to upload

Definition at line 256 of file dropboxSync.m.

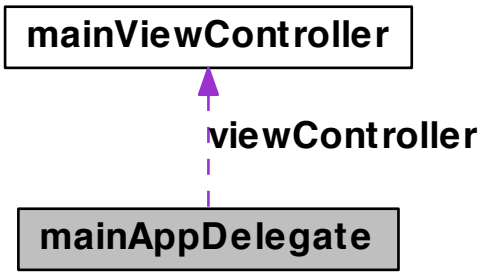
The documentation for this class was generated from the following files:

- dropboxSync.h
- dropboxSync.m

5.10 mainAppDelegate Class Reference

Handles application creation and logic flow.

Collaboration diagram for mainAppDelegate:



Public Member Functions

- (void) - alertView:clickedButtonAtIndex:

  Delegate for overwrite-database alert popup.

Protected Attributes

- UIWindow * **window**
- mainViewController * **viewController**

Properties

- IBOutlet UIWindow * window

  Application's main window.
- UINavigationController * navController

  Navigation controller (UI bar)
- IBOutlet mainViewController * viewController

  Application's main view controller.
- codeScanner * scanner

  Application's barcode scanning logic.
- databaseManager * dbManager

  Application's global database manager.
- dropboxSync * dropbox

  Application's dropbox synchronization manager.
- id< customerProtocol > customer

  Class instance that handles getting customer info from database.

- NSURL * newDatabaseFileUrl

  URL of new database file from external application.

5.10.1 Detailed Description

Handles application creation and logic flow.

Creates a mainViewController, the main interface for the application. Also stores the application-global class instances.

Definition at line 33 of file mainAppDelegate.h.

5.10.2 Member Function Documentation

5.10.2.1 - (void) alertView: dummy(UIAlertView *) *alertView* clickedButtonAtIndex:(NSInteger) *buttonIndex*

Delegate for overwrite-database alert popup.

If application is given a database file, it prompts user asking if he wants to overwrite the existing database. This delegate method is called with the user response to that question.

If user clicked cancel, nothing should happen. If user clicked OK, the new database should be copied over the existing one.

Parameters

alertView	Alert that called this delegate
buttonIndex	Button the user clicked

Definition at line 144 of file mainAppDelegate.m.

Here is the call graph for this function:

The documentation for this class was generated from the following files:

- mainAppDelegate.h
- mainAppDelegate.m

5.11 mainViewController Class Reference

Main view controller during normal use.

Properties

- UIImage * **wheelImage**
- cameraView * **cameraView**

5.11.1 Detailed Description

Main view controller during normal use.

This view controller simply initializes a new rootView and sets it as the fullscreen view.

Definition at line 26 of file mainViewController.h.

The documentation for this class was generated from the following file:

- mainViewController.h

5.12 numberInputVC Class Reference

Public Member Functions

- (id) - **initWithExistingNumber:withUserData:**

Properties

- UITextField * **textField**
- id **delegate**
- id userData

  Identifier data passed back to delegate after an event occurs.

5.12.1 Detailed Description

Definition at line 27 of file numberInputVC.h.

The documentation for this class was generated from the following files:

- numberInputVC.h
- numberInputVC.m

5.13 <NumberInputVCProtocol> Protocol Reference

Public Member Functions

- (void) - **numberInputView:withUserData:updatedNumber:**

5.13.1 Detailed Description

Definition at line 42 of file numberInputVC.h.

The documentation for this protocol was generated from the following file:

- numberInputVC.h

5.14 phoneInputVC Class Reference

Displays a single, editable text-field and keyboard.

Public Member Functions

- (id) - initWithExistingText:withUserData:

  Initialize view controller with existing text and identifying data.

Properties

- UITextField * textField

  The editable text field within the displayed table cell.
- id delegate

  Delegate implementing phoneInputVCProtocol.
- id userData

  Identifier data passed back to delegate after an event occurs.

5.14.1 Detailed Description

Displays a single, editable text-field and keyboard.

Definition at line 27 of file phoneInputVC.h.

5.14.2 Member Function Documentation

5.14.2.1 - (id) initWithExistingText: dummy(NSString*) *initText* withUserData:(id) *initUserData*

Initialize view controller with existing text and identifying data.

Parameters

initText	Text to pre-fill text field with
initUserData	Data passed back to delegate. Used to ID caller.

Returns

Initialized instance

Definition at line 43 of file phoneInputVC.m.

The documentation for this class was generated from the following files:

- phoneInputVC.h
- phoneInputVC.m

5.15 <phoneInputVCProtocol> Protocol Reference

Public Member Functions

- (void) - phoneInputView:withUserData:updatedText:

  Protocol describing how phoneInputVC informs delegate of text change.

5.15.1 Detailed Description

Definition at line 45 of file phoneInputVC.h.

The documentation for this protocol was generated from the following file:

- phoneInputVC.h

5.16 rootView Class Reference

Main view during normal use, divides screen into two subviews.

Public Member Functions

- (id) - **initWithFrame:**
- (void) - **dealloc**
- (void) - causePulseInMainThread

  Call pulseOverlay: on main thread.
- (void) - pulseOverlay

  Briefly flashes screen white.
- (void) - **disableView**
- (void) - **enableView**

Properties

- UIView * **disabledOverlayView**

5.16.1 Detailed Description

Main view during normal use, divides screen into two subviews.

This view is displayed full-screen during normal (non-administrative) operation, and divides itself into two subviews. The top subview, 1/4th of the screen, is given to a cameraView. The bottom 3/4ths of the screen is given to a customerInfoView.

This view also handles the user input that launches the administration UI as a modal view.

Definition at line 34 of file rootView.h.

5.16.2 Member Function Documentation

5.16.2.1 - (void) causePulseInMainThread

Call pulseOverlay: on main thread.

Since pulseOverlay: requires graphics redraws, it must be run on the main thread. But since it is called by a notification that is not necessarily running on the main thread, this function gets called by the notification and scheduls pulseOverlay: to be called later on the main thread.

Definition at line 115 of file rootView.m.

5.16.2.2 - (void) pulseOverlay

Briefly flashes screen white.

Visual indicator when a barcode is successfully scans, overlays the entire screen white that rapidly fades in and out in opacity. Result is a quick white flash that only semi-obscures the screen, and provides feedback that a scan succeeded.

Definition at line 130 of file rootView.m.

The documentation for this class was generated from the following files:

- rootView.h
- rootView.m

5.17 textFieldInputVC Class Reference

Displays a single, editable text-field and keyboard.

Public Member Functions

- (id) - initWithExistingText:withUserData:

  Initialize view controller with existing text and identifying data.

Properties

- UITextField * textField

  The editable text field within the displayed table cell.
- id delegate

  Delegate implementing textInputVCProtocol.
- id userData

  Identifier data passed back to delegate after an event occurs.

5.17.1 Detailed Description

Displays a single, editable text-field and keyboard.

This is a full UITableViewController that displays a single cell with an editable text field in it. It has a permanently displayed virtual keyboard, too.

Expected usage is to have a table view that shows non-editable text fields. When a user selects one of these fields, it creates a textFieldInputVC view and shows it with animation. The user enters or edits text data, and then closes this view. The launching view should then update its text content with the result.

This view is expected to be pushed on a navigation controller's stack, and will subsequently pop itself off the stack when the user finishes.

The header for this class also defines a protocol that it uses for returning the result of the user's input. A view that uses this class is expected to implement textInputVCProtocol.

Definition at line 27 of file textFieldInputVC.h.

5.17.2 Member Function Documentation

5.17.2.1 - (id) initWithExistingText: dummy(NSString*) *initText* withUserData:(id) *initUserData*

Initialize view controller with existing text and identifying data.

Parameters

initText	Text to pre-fill text field with
initUserData	Data passed back to delegate. Used to ID caller.

Returns

Initialized instance

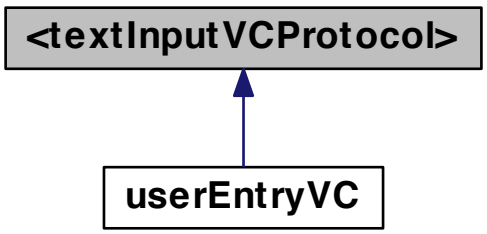
Definition at line 60 of file textFieldInputVC.m.

The documentation for this class was generated from the following files:

- textFieldInputVC.h
- textFieldInputVC.m

5.18 <textInputVCProtocol> Protocol Reference

Inheritance diagram for <textInputVCProtocol>:



Public Member Functions

- (void) - textInputView:withUserData:updatedText:

  Protocol describing how textFieldInputVC informs delegate of text change.

5.18.1 Detailed Description

Definition at line 45 of file textFieldInputVC.h.

5.18.2 Member Function Documentation

5.18.2.1 - (void) textInputView: dummy(textFieldInputVC *) *textView* withUserData:(id) *data* updatedText:(NSString *) *text*

Protocol describing how textFieldInputVC informs delegate of text change.

A textFieldInputVC is created and displayed by another view. That view gets informed of changes to the text via the delegate callback methods specified in textInputVCProtocol.

Expected usage is to have a view controller that implements textInputVCProtocol. It will create a textFieldInputVC view, set itself as the delegate, and push the text field view onto the view controller stack. When the user is finished, the methods defined in this protocol will be called on the originating view.

The documentation for this protocol was generated from the following file:

- textFieldInputVC.h

5.19 userAdminVC Class Reference

Public Member Functions

- (id) - initWithDbFile:

  Initialize a new instance with the given database.
- (void) - readRowsFromDb

  Reads names/barcodes of all customers from the database.
- (void) - **disableTableViews**
- (void) - **enableTableViews**

Properties

- NSString ∗ dbFile

  Full path to database file.
- NSArray ∗ allRows

  Local copy of names/barcodes of all customers in the database.
- NSArray ∗ searchRows

  Copy of customers in allRows who match the current search terms.
- UISearchBar ∗ searchBar

  Search bar UI element.
- BOOL doNotSaveDatabase

  Set to tell controller not to save database when it disappears.
- BOOL **searchResultsActive**
- NSString ∗ **searchString**
- UISearchDisplayController ∗ **searchController**
- UIView ∗ **overlay**
- UIView ∗ **searchOverlay**

5.19.1 Detailed Description

Definition at line 27 of file userAdminVC.h.

5.19.2 Member Function Documentation

5.19.2.1 - (id) initWithDbFile: dummy(NSString∗) *db*

Initialize a new instance with the given database.

Initializer stores the database, reads all the customers from the database, and creates a search bar and search controller.

Parameters

db	Database file to administer

Returns

Initialized instance of class

Definition at line 99 of file userAdminVC.m.

5.19.2.2 - (void) readRowsFromDb

Reads names/barcodes of all customers from the database.

Reads names/barcodes of all the customers, and stores them in a global variable that the cells will use to populate themselves. They are sorted by a guess at the last name (the last word of the name field).

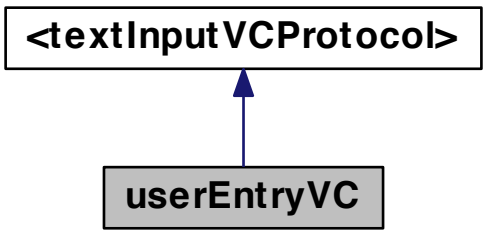
Definition at line 157 of file userAdminVC.m.

The documentation for this class was generated from the following files:

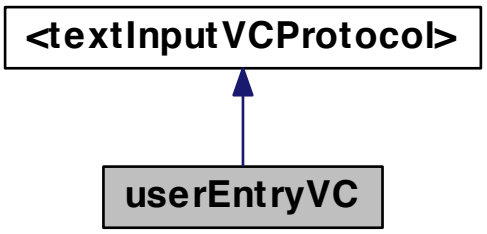
- userAdminVC.h
- userAdminVC.m

5.20 userEntryVC Class Reference

Inheritance diagram for userEntryVC:



Collaboration diagram for userEntryVC:



Public Member Functions

- (id) - initWithStyle:withDbFile:withBarcode:

  Initialize customer entry UI form.
- (NSDictionary ∗) - rowMetadataFromIndexPath:

  Get metadata describing what information should be stored at given cell.
- (NSMutableArray ∗) - **initContent**

Properties

- NSString ∗ dbFile

  Full path to database file.
- NSString ∗ barcode

  Barcode of the customer currently being created/viewed.
- NSMutableArray ∗ content

  Local copy of all information about this customer from the database.

5.20.1 Detailed Description

Definition at line 27 of file userEntryVC.h.

5.20.2 Member Function Documentation

5.20.2.1 - (id) initWithStyle: dummy(UITableViewStyle) *style* withDbFile:(NSString∗) *db* withBarcode:(NSString∗) *code*

Initialize customer entry UI form.

Parameters

style	Valid iOS UITableView style
db	Database file to operate on
barcode	Barcode of customer to edit, or nil for new customer

Returns

New instance of class

Definition at line 63 of file userEntryVC.m.

5.20.2.2 - (NSDictionary ∗) rowMetadataFromIndexPath: dummy(NSIndexPath ∗) *indexPath*

Get metadata describing what information should be stored at given cell.

Parameters

indexPath	Section and row of cell to get info for

Returns

Dictionary from customerDefinition that describes this cell's data

Definition at line 307 of file userEntryVC.m.

The documentation for this class was generated from the following files:

- userEntryVC.h
- userEntryVC.m

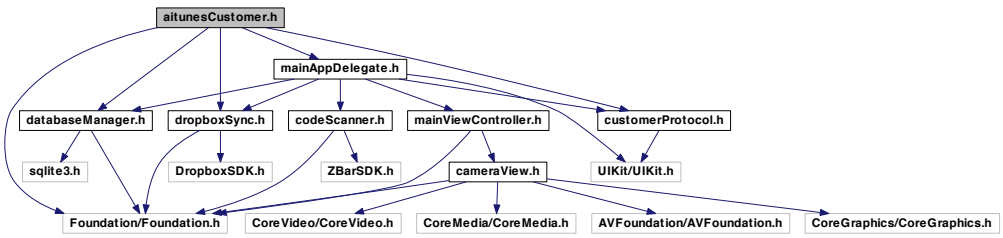
Chapter 6

File Documentation

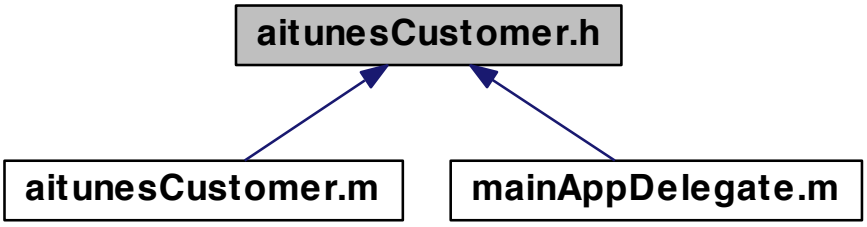
6.1 aitunesCustomer.h File Reference

```
#import <Foundation/Foundation.h>
#import "customerProtocol.h"
#import "databaseManager.h"
#import "dropboxSync.h"
#import "mainAppDelegate.h"
```

Include dependency graph for aitunesCustomer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class aitunesCustomer

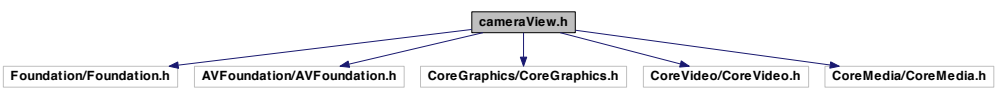
6.1.1 Detailed Description

Definition in file aitunesCustomer.h.

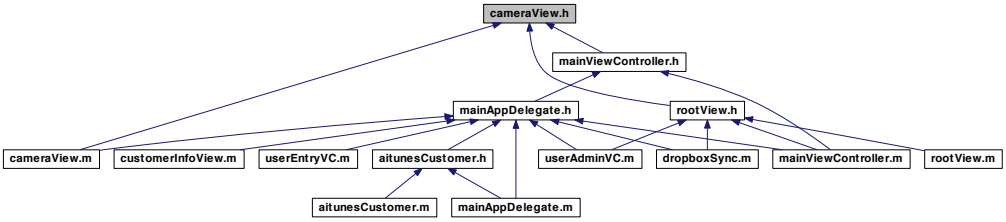
6.2 cameraView.h File Reference

```
#import <Foundation/Foundation.h>
#import <AVFoundation/AVFoundation.h>
#import <CoreGraphics/CoreGraphics.h>
#import <CoreVideo/CoreVideo.h>
#import <CoreMedia/CoreMedia.h>
```

Include dependency graph for cameraView.h:



This graph shows which files directly or indirectly include this file:



Classes

- class cameraView

  Cropped view of rear camera video stream.

Defines

- #define VIDEO_ENLARGEMENT_FACTOR 1.0666

6.2.1 Detailed Description

Definition in file cameraView.h.

6.2.2 Define Documentation

6.2.2.1 #define VIDEO_ENLARGEMENT_FACTOR 1.0666

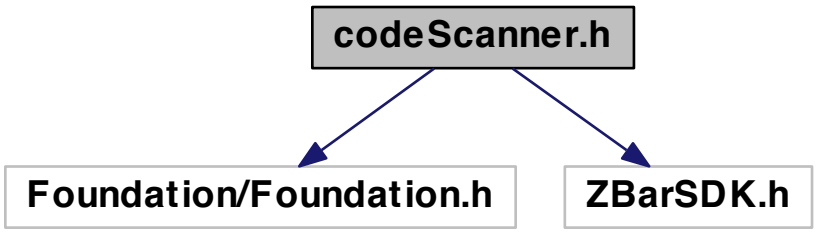
Factor to scale video frame by to make it full screen, because iPad screen is 768px wide and the video frame is 720px wide. 768/720 ~= 1.0666

Definition at line 34 of file cameraView.h.

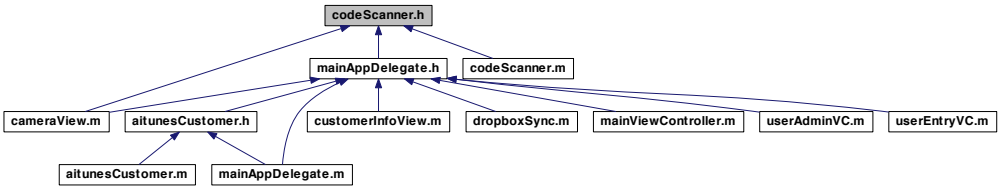
6.3 codeScanner.h File Reference

```
#import <Foundation/Foundation.h>
#import "ZBarSDK.h"
```

Include dependency graph for codeScanner.h:



This graph shows which files directly or indirectly include this file:



Classes

- class codeScanner

  Handles scanning images for barcodes.

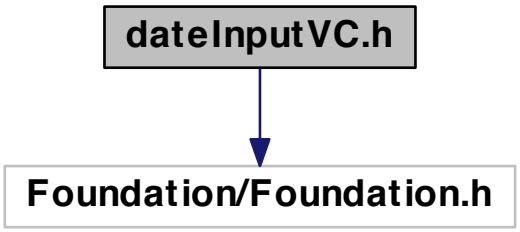
6.3.1 Detailed Description

Definition in file codeScanner.h.

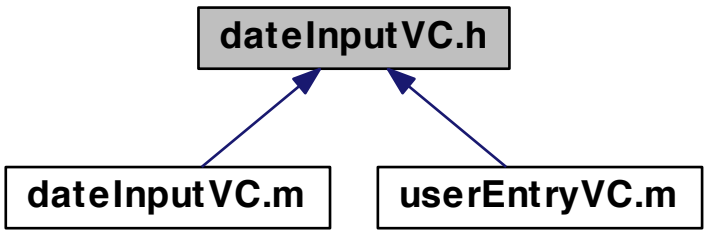
6.4 dateInputVC.h File Reference

```
#import <Foundation/Foundation.h>
```

Include dependency graph for dateInputVC.h:



This graph shows which files directly or indirectly include this file:



Classes

- class dateInputVC
- protocol <dateInputVCProtocol>

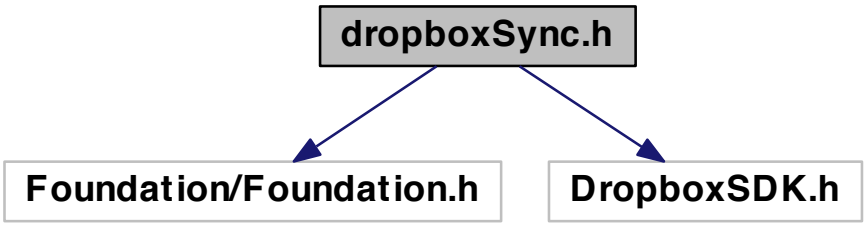
6.4.1 Detailed Description

Definition in file dateInputVC.h.

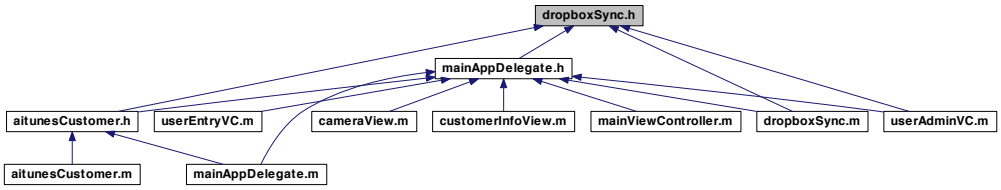
6.5 dropboxSync.h File Reference

```
#import <Foundation/Foundation.h>
#import "DropboxSDK.h"
```

Include dependency graph for dropboxSync.h:



This graph shows which files directly or indirectly include this file:



Classes

- class dropboxSync

6.5.1 Detailed Description

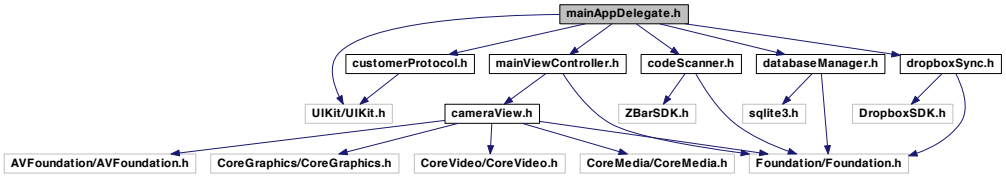
Definition in file dropboxSync.h.

6.6 mainAppDelegate.h File Reference

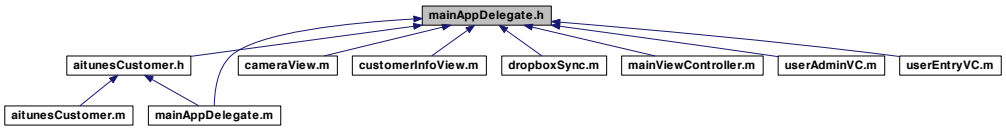
```
#import <UIKit/UIKit.h>
```

```
#import "mainViewController.h"
#import "codeScanner.h"
#import "databaseManager.h"
#import "customerProtocol.h"
#import "dropboxSync.h"
```

Include dependency graph for mainAppDelegate.h:



This graph shows which files directly or indirectly include this file:



Classes

- class mainAppDelegate

  Handles application creation and logic flow.

Defines

- #define **ASE_VERSION** @"1.0"

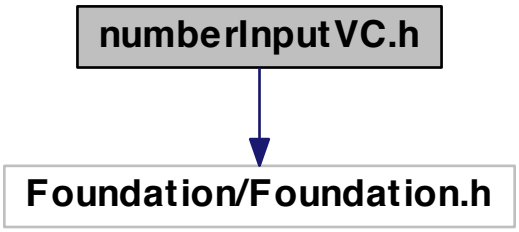
6.6.1 Detailed Description

Definition in file mainAppDelegate.h.

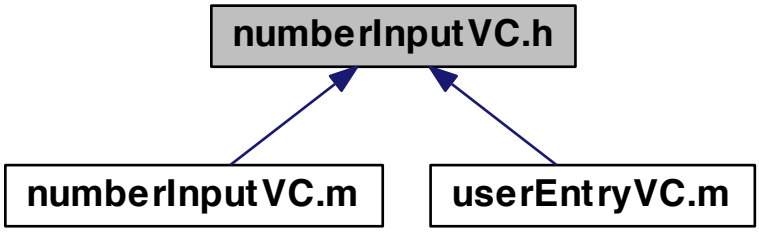
6.7 numberInputVC.h File Reference

```
#import <Foundation/Foundation.h>
```

Include dependency graph for numberInputVC.h:



This graph shows which files directly or indirectly include this file:



Classes

- class numberInputVC
- protocol <NumberInputVCProtocol>

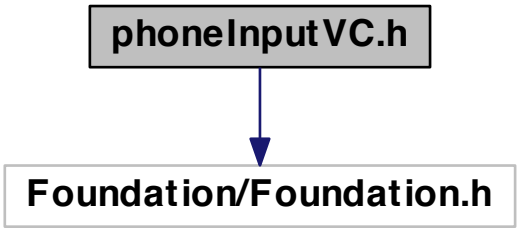
6.7.1 Detailed Description

Definition in file numberInputVC.h.

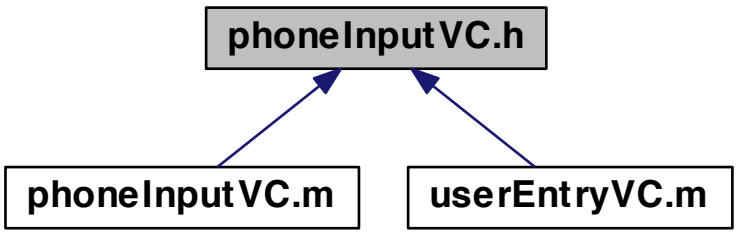
6.8 phoneInputVC.h File Reference

```
#import <Foundation/Foundation.h>
```

Include dependency graph for phoneInputVC.h:



This graph shows which files directly or indirectly include this file:



Classes

- class phoneInputVC

  Displays a single, editable text-field and keyboard.
- protocol <phoneInputVCProtocol>

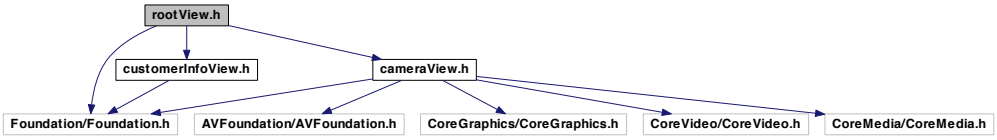
6.8.1 Detailed Description

Definition in file phoneInputVC.h.

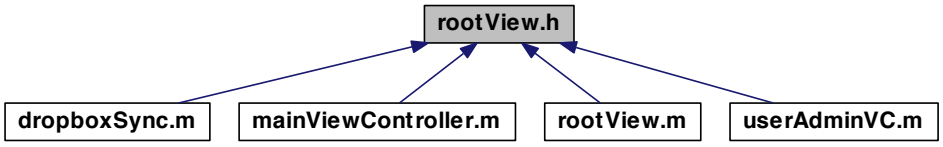
6.9 rootView.h File Reference

```
#import <Foundation/Foundation.h>
#import "cameraView.h"
#import "customerInfoView.h"
```

Include dependency graph for rootView.h:



This graph shows which files directly or indirectly include this file:



Classes

- class rootView

  Main view during normal use, divides screen into two subviews.

Defines

- #define CAMERA_FRAME_DIVIDER 4

  What portion of screen is dedicated to camera frame (1/N of screen)
- #define CUSTOMER_FRAME_DIVIDER 1.333

  What portion of screen is dedicated to customer info (1/N of screen)

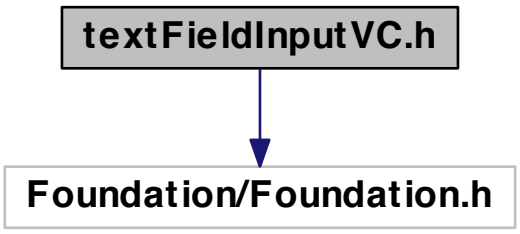
6.9.1 Detailed Description

Definition in file rootView.h.

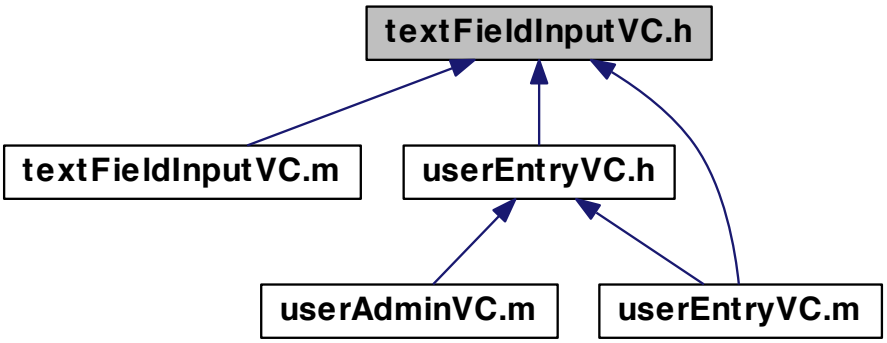
6.10 textFieldInputVC.h File Reference

```
#import <Foundation/Foundation.h>
```

Include dependency graph for textFieldInputVC.h:



This graph shows which files directly or indirectly include this file:



Classes

- class textFieldInputVC

  Displays a single, editable text-field and keyboard.

- protocol <textInputVCProtocol>

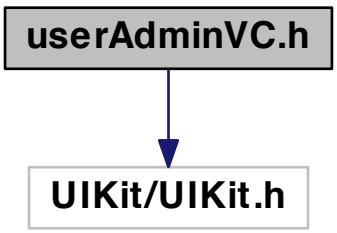
6.10.1 Detailed Description

Definition in file textFieldInputVC.h.

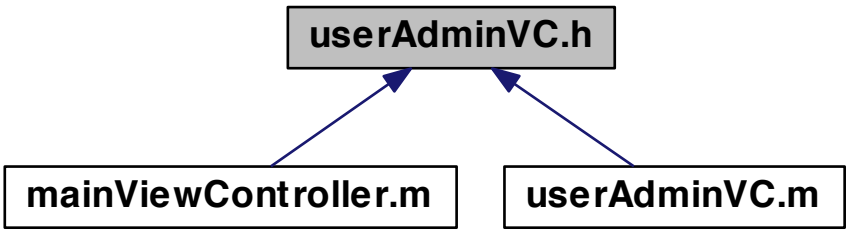
6.11 userAdminVC.h File Reference

```
#import <UIKit/UIKit.h>
```

Include dependency graph for userAdminVC.h:



This graph shows which files directly or indirectly include this file:



Classes

- class userAdminVC

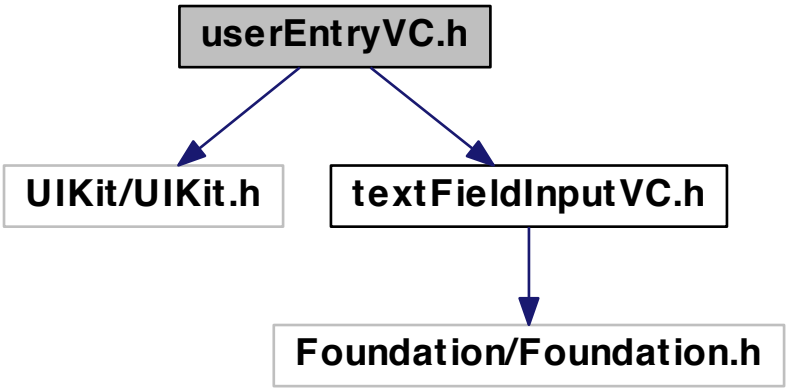
6.11.1 Detailed Description

Definition in file userAdminVC.h.

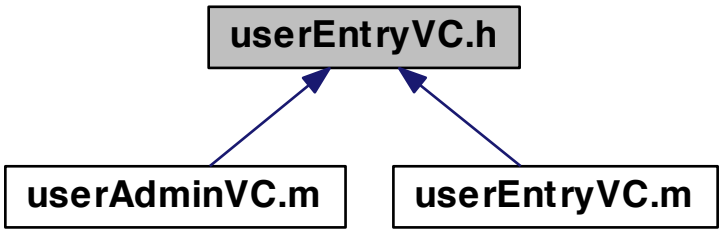
6.12 userEntryVC.h File Reference

```
#import <UIKit/UIKit.h>
#import "textFieldInputVC.h"
```

Include dependency graph for userEntryVC.h:



This graph shows which files directly or indirectly include this file:



Classes

- class userEntryVC

6.12.1 Detailed Description

Definition in file userEntryVC.h.

Index