Team 8: Technical Documentation

Kerridge Commercial Systems Application

Document Breakdown

This document is the Technical Documentation that will be available alongside the completion of the Application for Kerridge Commercial Systems. It will also be available alongside the Manual Documentation.

- The Technical Documentation can be viewed by experienced users to find out more about what is happening "behind the scenes" of the application. It will not contain all information/code definitions, only the ones that are deemed important by leaders in Team 8's programming team.
- The Manual Documentation depicts how to use each part of the application, accompanied by up-to-date screenshots of the app for visual conformation of the actions that the user can perform.

Both documents will allow users to deeper understand how the application works and both will be available from the website under the User Manual and Developer Manual tabs.

Any time a piece of code is referred to, such as a variable, function name, class name or library and etc, they will be denoted by an underline.

It is worth noting that there are a few functions that will be constantly mentioned throughout the document. However, they must be mentioned individually because their use within the application varies and is not always the same. These functions include: onCreate, onClick, onResponse and onFailure.

The onCreate function is responsible for initialising the activities for each of the screens for the application; including each of their functions and the information that each one will provide.

The onClick function refers to an event which occurs when an object on screen is interacted with by the user. Functions can be set to activate when these events are triggered.

The onResponse and onFailure events occur in unison. The onResponse function will trigger when information is successfully passed to it. From there, the application can decide if the information is correct or incorrect and which response to reply with. The onFailure event will trigger if an error has occurred (which is passed through a parameter Throwable t in most cases).

Finally, an Important Information section has been added to the end of this document which contains links to the classes that were not written by the programming team but in fact used from the Android Studio API. This section also contains more information on AppCompatActivity, which contains the 4 methods described above.

Document Information

Document Name	Technical Documentation
Created By	Conor Lambert
Creation Date	15th April 2019
Version	1.9.4
Version Date	28th April 2019

Version Breakdown

Version	Description of Additions/Changes
1.0	Creation of document and creation of a simple template to follow for class and function definitions.
1.1	Inserted a cover page relevant to the project. Continued breakdown of the LoginScreen class – updating variable and function names where necessary.
1.2	Change vocabulary found throughout the document to match the formal jargon used throughout Android Studio. Edited the breakdown of the LoginScreen class to meet the updated class from GitHub.
1.3	Updated information in the document to match version on GitHub. Added Note.java and NoteAdapter.java to the document. Completed the breakdown on the following sections: ApiUnprotected.java, AuthenticationInteceptor.java, Calendar.java and Client.java. Decided to add a variables section to each breakdown section.
1.4	Continued to update the information that has changed on GitHub. Added an Important Information section to the document as well as its definition in the document Breakdown. Completed the ClientAdapter.java and ClientListScreen.java sections.
1.5	Implemented the sections for CreateNewNote.java and DatePickerFragment.java ready for completion. Completed the following sections: CreateNewNote.java, DatePickerFragment.java, LoginDetails.java, LoginResponse.java. Updated the LoginScreen.java to make sure it was fully up to date.
1.6	Finished the following sections: CalendarScreen.java, Meeting.Java, MeetingAdapter.java and MeetingInformation.java. Updated the document information to ensure that all the information was correct and accurate. Tweaked sections for the document to make sure that the code descriptions were also up to date.
1.7	Added the sections NoteDetails.java, RetrieveTweets.java, TimePickerFragment.java and UpdateNote.java ready for completion.

	Completed the sections MeetingNotes.java, Note.java, NoteAdapter.java and NoteDetails.java.
1.8	Completed the sections ScheduleMeetings.java, ServiceGenerator.java, SplashScreen.java, TimePickerFragment.java, UpcomingMeetings.java and UpdateNote.java
1.9	Completed the final sections of the document ready for the submission.
1.9.1	Updated Document information, including minor changes to the code. Added one small section for a new class.
1.9.2	Further changes to the document to meet the changes made to the application. Removed the NoteDetails.java section for the updated CreateNoteDetails section.
1.9.3	Minor tweaks to improve spelling, punctuation and grammar. Updated information to ensure accuracy between this document and the code for the application.
1.9.4	Minor tweaks to meet the updated code on the GitHub. Removed EditClientInfoActivity.java section as now obsolete.

Contents Page (To be completed for Final Version)

Class Breakdown: ApiConst.java

Class Name	ApiConst
Constructor(s)	None
Function(s) to be described	None
Variables	String API_BASE String API_LOGIN String API_CLIENT String API_MEETING String API_NOTE String API_IMAGES

Though this class provides no functions for use, that does not limit its usefulness. The idea of this class is to provide the constants for the application development and contains 7 constant variables that contain locations to other pages and their activities.

For example, the location for the login screen is made up of the *API_BASE + API_LOGIN* equating to the concatenation of the two strings provided by the variable assignments.

The API_IMAGES constant is set to API_BASE + "static/images/" in case the location contained within API_BASE is changed (which would mean that only API_BASE would be changed and not API_IMAGES).

Interface Breakdown: ApiProtected.java

Interface Name	ApiProtected
Constructor(s)	None
Function(s) to be described	getClients getMeetings createMeeting updateMeeting getNotes createNote updateNote
Variables	None

This class makes use of the Retrofit library. For more information on Retrofit, see the Important Information section for a link to the documentation.

ApiProtected is used when a login is completed alongside a correct authentication token. If there is no valid token, then ApiUnprotected is called (as it would be a catastrophe if someone had access to information that they shouldn't have – especially if it was sensitive).

Function Breakdown: getClients, getMeetings, createMeeting, updateMeeting, getNotes, createNote, updateNote

Function Name	Return Type	Parameters	Variables
getClients	Call<List<Client>>	None	None
getMeetings	Call<List<Meeting>>	None	None
createMeeting	Call<Void>	CreateMeetingBody createMeetingBody	None
updateMeeting	Call<Void>	UpdateMeetingBody updateMeetingBody	None

getNotes	Call<List<Note>>	Integer meetingId	None
createNote	Call<Note>	CreateNoteBody createNoteBody	None
updateNote	Call<Note>	UpdateNoteBody updateNoteBody	None

An @GET is included in the Retrofit imports and represents a HTTP GET request for information from ApiConst. In the case of getClients the @GET request is for the API_CLIENT constant and for getMeetings it is for the API_MEETING; both included in ApiConst.java.

The getNotes function is slightly different as it takes one parameter; this parameter is meetingId and its value is returned from a HTTP Query request from Retrofit. The postNote method will take the createNoteDetails and pass them through to save to the database under the meetingId that contains it. This also includes updateNote for any notes that are pre-existing and are expecting changes to be made.

As this is an interface, these methods are acting as placeholders for any class that would implement this interface. Should they need defining, this will be done by the class that needs to use them. These methods cannot be accessed unless the class implements this interface.

Interface Breakdown: ApiUnprotected.java

Interface Name	ApiUnprotected
Constructor(s)	None
Function(s) to be described	userLogin
Variables	None

ApiUnprotected is used when a login is completed without a valid authentication token. Without a valid token, the user cannot access information from the database to stop any sensitive data from being stolen by an unwanted user.

Function Breakdown: userLogin

Function Name	Return Type	Parameters	Variables
userLogin	Call<LoginResponse	LoginBody loginBody	None

ApiUnprotected is an interface, and the methods in the interface act as placeholders. The interface must be implemented by another class if that class wants to use the methods provided.

The userLogin function gets loginDetails passed to it to then uses said data to log the user in. The LoginDetails are made up of email and password.

Class Breakdown: AuthenticationInterceptor.java

Class Name	AuthenticationInterceptor
Constructor(s)	Yes
Function(s) to be described	intercept
Variables	String authToken

AuthenticationInterceptor acts as a middleman between the user logging in and the database holding notes and other information. It will take the token from the login request and use it to build a connection to the database. This token will then be known as the user's authentication token.

Function Breakdown: intercept

Function Name	Return Type	Parameters	Variables
Intercept	Response	Chain chain	(Request) original (Request.builder) builder (Request) request

Class Breakdown: CalendarActivity.java

Class Name	CalendarActivity
Constructor(s)	None
Function(s) to be described	onCreate displayMeetingForDay onDayClick onMonthScroll
Variables	SharedPreferences sharedPreferences CompactCalendarView compactCalendar RecyclerView rv RecyclerAdapter.Adapter adapter RecyclerView.LayoutManager lm TextView txtMonth SimpleDateFormat dateFormat

This is where the Calendar screen is built from. The Calendar screen will allow the user to click on specific days and select the meetings and notes that they have on those days. From there they can add, edit or delete notes. The onCreate function in this instance will highlight the current date that the user is viewing the screen on. Furthermore, this class contains details about what style the Calendar is in. If the user swipes left or right on the screen, it will correctly update the information.

This class uses a GitHub library for the Calendar itself. A link to the GitHub will be provided in Important Information. The import used is CompactCalendarView.

Function Breakdown: onCreate, displayMeetingForDay

Function Name	Return Type	Parameters	Variables
onCreate	Void	Bundle savedInstanceState	None
displayMeetingForDay	Void	Date dateClicked	(List<Event>) events (List<Meeting>) selected (Meeting) temp

As aforementioned, onCreate is responsible for creating the screen that the calendar can be found on. A full description of what onCreate does can be found in the document breakdown section. It is from this function that the screen will appear as it does, including location of the Calendar and any other elements that may be added. The onClick event in this class will take information from UpcomingMeetings and display it on the day that has been clicked (as long as the information is corresponding to that day).

Function Breakdown: onDayClick, onMonthScroll, getMeetings, onResponse, onFailure

Function Name	Return Type	Parameters	Variables
onDayClick	Void	Date dateClicked	(List<Event>) events (List<Meeting>) selected
onMonthScroll	Void	Date firstDayOfNewMonth	None

The onDayClick function is solely responsible for making sure that the correct information is displayed to the user when they click on any day in the calendar. When a day is clicked, the information is saved to events (through the function getEvents(dateClicked) which is provided by the library) which will then be used to display the events on that day. The selected variable will be filled with new meeting objects corresponding to the events on the day that was clicked, which allows for a new MeetingAdapter to be created.

The onMonthScroll is responsible for making sure that the correct data is displayed when the user swiped left or right on the screen. When this happens, the activity on the screen is updated resulting in getMeetings being called.

Function Name	Return Type	Parameters	Variables
getMeetings	Void	None	(Call<List<Meeting>>) call
onResponse	Void	Call<List<Meeting>> call Response<List<Meeting>> response	(List<Meeting>) ml (Event) tempEvent
onFailure	Void	Call<List<Meeting>> call Throwable t	None

The getMeetings function will be used to keep the meetings information on the calendar screen up to date. It is also responsible for the appearance of the events under each day on the screen. If there are no problems with retrieving the information for upcoming meetings, the onResponse will be used to loop through the meetings to create events to display on the calendar screen. However, if anything goes wrong with retrieving the information then onFailure will display an error message to the user.

Class Breakdown: Client.java

Class Name	Client
Constructor(s)	Yes
Function(s) to be described	Various Getters and Setters createFromParcel newArray describeContents writeToParcel
Variables	Integer id String name String photo Parcelable.Creator CREATOR

This class provides all the information tied to a client. The information can be retrieved through the getters that are provided and edit the information by using the setters in the class as well.

Function Breakdown: createFromParcel, newArray

Function Name	Return Type	Parameters	Variables
createFromParcel	Parcelable.Creator	Parcel in	None
newArray	Client [] (Array)	int size	None

These two methods are tied to the CREATOR variable as they are included when Parcelable.Creator is called to initialise it. The createFromParcel method will create a new client object from the in parameter and the newArray method will create an array to return equal to the number of clients that are required to fill it.

Function Breakdown: describeContents, writeToParcel

Function Name	Return Type	Parameters	Variables
describeContents	int	None	None
writeToParcel	Void	Parcel dest int flags	None

The describeContents method is required by Retrofit and in the terms of this application there is no other instance that is required other than for it to return 0. The writeToParcel method will take a Parcel input (dest) and write the id and name to the Parcel that have been provided.

Class Breakdown: ClientAdapter.java

Class Name	ClientAdapter
Constructor(s)	Yes
Sub Class Name	MyViewHolder
Sub Class Constructor(s)	Yes
Function(s) to be described	onCreateViewHolder onBindViewHolder getItemCount
Variables	List<Client> clientList Context context

This class extends the RecyclerView abstract class. A link to the official API documentation can be found in the Important Information section of this document for more information on the class itself.

The constructor for this class will create a new ClientAdapter object containing a clientList and context provided through its parameters cl and context respectively.

Function Breakdown: onCreateViewHolder, onBindViewHolder, getItemCount

Function Name	Return Type	Parameters	Variables
onCreateViewHolder	MyViewHolder	ViewGroup parent int viewType	(View) itemView
onBindViewHolder	Void	MyViewHolder holder int position	(Client) client
getItemCount	int	None	None

Both the onCreateViewHolder and onBindViewHolder methods are defined in the RecyclerView class (see Important Information). The onCreateViewHolder method is called when createViewHolder is used from RecyclerView and is used to create a new ViewHolder object with the parameters provided. The onBindViewHolder method is internally called when bindViewHolder is called from RecyclerView to update the ViewHolder contents with the item at the given position.

The getItemCount method will return the size of the clientList variable when called.

Sub Class Breakdown: ViewHolder

Sub Class Name	ViewHolder
Sub Class Constructor(s)	Yes
Variables	TextView ClientID TextView clientName ImageView clientPhoto

This sub class contains a constructor that will create a new MyViewHolder object when called. The clientId and clientName variables will be set when findViewById returns a value for each one.

Class Breakdown: ClientListActivity.java

Class Name	ClientListActivity
Constructor(s)	None
Function(s) to be described	onCreate populateClientList onResponse onFailure
Variables	RecyclerView rv SharedPreferences sharedPreferences LinearLayoutManager lm RecyclerView.Adapter adapter

Function Breakdown: onCreate, populateClientList, onResponse, onFailure

Function Name	Return Type	Parameters	Variables
onCreate	Void	Bundle savedInstanceState	None
populateClientList	Void	None	(ApiProtected) ApiProtected (Call<List<Client>>) call
onResponse	Void	Call<List<Client>> call Response<List<Client>> response	None
onFailure	Void	Call<List<Client>> call Throwable t	None

The onCreate method is called to create the screen and set up its functions and variables. In this class it will also call populateClientList. The populateClientList method will create a new ApiProtected object, utilizing the sharedPreferences variable that was set up along with the class. This method will create a new Call<List<Client>> object, call, which has the returned value of ApiProtected.getClients() (see ApiProtected.java). The onResponse and onFailure methods are used when data is retrieved from the

database. If an error should occur, onFailure will be called to display an error / warning message for the user.

Class Breakdown: ClientProfileActivity.java

Class Name	ClientProfileActivity
Constructor(s)	Yes
Function(s) to be described	onCreate genNewsList
Variables	ArrayList<twitter4j.Status> tweetList

This class extends AppCompatActivity which contains methods like onCreate. See the Important Information section for a link to the documentation page for information on this class.

Function Breakdown: onCreate, genNewsList

Function Name	Return Type	Parameters	Variables
onCreate	Void	Bundle savedInstanceState	None
genNewsList	Void	None	(RetrieveTweets) rt

The onCreate function is responsible for the initialisation of this activity. Upon it calling, it will finish by calling genNewsList which is responsible for retrieving the necessary tweets to display.

Class Breakdown: CreateMeetingBody.java

Class Name	CreateMeetingBody
Constructor(s)	Yes
Function(s) to be described	Various Getters and Setters
Variables	Date startDate Date endDate Integer clientId String location

All CreateMeetingBody objects contain startDate, endDate, clientId and location. The class provides many getters and setters to allow the application to retrieve any information that is required and also change it when necessary. There is also toString method that allows for a string format of all the information to be created.

Class Breakdown: CreateNoteActivity.java

Class Name	CreateNoteActivity
Constructor	None
Function(s) to be described	onCreate onClick saveNote onResponse onFailure
Variables	SharedPreferences sharedPreferences TextView createNoteTitles

	EditText createNoteBody EditText createNoteHeading Meeting meeting

Function Breakdown: onCreate, onClick, saveNote, onResponse, onFailure

Function Name	Return Type	Parameters	Variables
onCreate	Void	Bundle savedInstanceState	(Button) saveNoteButton
saveNote	Void	None	(String) heading (String) contents (int) meetingId (CreateNoteDetails) createNoteDetails (ApiProtected) apiProtected
onResponse	Void	Call<Note> call Response<Note> response	None
onFailure	Void	Call<Note> call Throwable t	None

The onCreate event for this class will set up the class as usual along with a new button, saveNoteButton, which utilises the function saveNote. The saveNote function has an onClick event listener attached to trigger whenever the new button is pressed. The three variables of the function (heading, contents and meetingId) describe the note, where number is the meeting ID that it corresponds to. Should either heading or contents be empty, an error will be set to tell the user what is wrong. However, if this is not the case then a new CreateNoteDetails object will be created which contains the three variables (heading, contents, meetingId). The onResponse and onFailure functions will let the user know if the note has been successfully posted to the database or not.

Class Breakdown: CreateNoteBody.java

Class Name	CreateNoteBody
Constructor	Yes
Function(s) to be described	None
Variables	String heading String contents int meeting

The CreateNoteBody class contains all the information contained within each note, such as the heading, its contents and the meeting number that the information corresponds to. Any new CreateNoteBody object contains all this information. A toString has been provided in the class which will format a simple string of all the information in one place.

Class Breakdown: LoginBody.java

Class Name	LoginBody
Constructor	Yes
Function(s) to be described	None

Variables	String username String password

This class simply handles the creation of new LoginDetails objects, which contain username and password which are passed through the respective parameters to the constructor. These objects are used throughout the application and its various classes. Calling toString on any LoginDetails object will provide an output of what details are contained within it.

Class Breakdown: LoginResponse.java

Class Name	LoginResponse
Constructor	No
Function(s) to be described	Various Getters and Setters
Variables	String msg String token

This class allows the Application to set the msg and token variables and return them to other parts of the application. The toString can also be used to output a String containing the information from the variables.

Class Breakdown: LoginActivity.java

Class Name	LoginActivity
Function(s) to be described	userLogin onResponse onFailure onCreate
Variables	SharedPreferences sharedPreferences EditText emailField EditText passwordField

This class is responsible for the creation of the login screen.

Function Breakdown: onCreate

Function Name	Return Type	Parameters	Variables
onCreate	Void	Bundle savedInstanceState	(SharedPreferences) sharedPreferences

The onCreate function in this instance will set the screen up and create a new button, loginButton, whose onClick event will trigger the userLogin function.

Function Breakdown: userLogin, onResponse and onFailure

Function Name	Return Type	Parameters	Variables
userLogin	Void	None	(String) email (String) password (LoginDetails Object) loginDetails (email, password) (ApiUnprotected Object) apiUnprotected

			(Call<LoginResponse>) call

The function is called when the user clicks the Login button with details in the email and password text boxes: in Android Studio these boxes are formally named *EditText* fields and it should be noted that the password box has an input type set to *"textPassword"*. If either box is empty when the Login button is pressed, the user will be informed that either an email and/or password is required to be entered before attempting to log in. On successfully entering the information, a new LoginBody object is created by sending the email and password to a constructor (see LoginDetails.java).

This new object now contains the user's details, which can be passed to the userLogin function within apiUnprotected and assigning its return value to the object call. The call.enqueue will then make use of two new inline functions:

Function Name	Return Type	Parameters	Variables
onResponse	Void	Call<LoginResponse> call Response<LoginResponse> response	(String) token
onFailure	Void	Call<LoginResponse> call Throwable t	None

For the onResponse function, if the login details are valid and there are no problems signing the user into their account on the application, as depicted by the if(response.isSuccessful()), the user will receive an on-screen message saying "Login Successful" and then move to the next screen on their account. However, if this is not the case, the user will be met with a message stating, "Login Unsuccessful", and therefore will be prompted to retry entering their details.

Should an error occur, the onFailure function will be used, stating the error that has occurred (which is contained within the Throwable t parameter).

Class Breakdown: Meeting.java

Class Name	Meeting
Constructor(s)	Yes
Function(s) to be described	Various Getters and Setters createFromParcel newArray writeToParcel describeContents
Variables	Integer id Date startDate Date endDate Client client Parcelable.Creator CREATOR

This class implements the Parcelable interface. For more information, see the link within the Important Information section to see more about the Parcelable interface in the Android Studio documentation.

The class contains many getters and setters that allow for the application to set the different parts of a meeting and receive the information to display for the user should the request it. Whenever the user edits a meeting, the setters will be used immediately to ensure that no data is lost within the process.

The constructor will take a parameter, in, and use it to fill the different fields of the meeting. These are id, date, duration and client. This will be further explained in the function breakdown.

Function Breakdown: createFromParcel, newArray, writeToParcel, describeContents

The functions createFromParcel and newArray are contained within the variable CREATOR.

Function Name	Return Type	Parameters	Variables
createFromParcel	Object	Parcel in	None
newArray	Meeting []	int size	None

CREATOR acts as the centre piece for creating the meetings. The createFromParcel function will create the meeting by passing the Parcel, in, to the meeting constructor. Furthermore, the newArray method will create an array of Meeting with a sized equal to the integer parameter passed in.

Function Name	Return Type	Parameters	Variables
writeToParcel	Void	Parcel dest int flags	None
describeContents	int	None	None

The writeToParcel function is responsible for creating a Parcel with the various functions provided by the Parcelable interface. The parcel created contains the variables that have been provided by the class (which are id, date, duration and client).

The describeContents method is required by Retrofit and in the terms of this application there is no other instance that is required other than for it to return 0.

A toString method has been provided so that all the information contained within Meetings.java can be printed out in a sensible fashion.

Class Breakdown: MeetingAdapter.java

Class Name	MeetingAdapter
Constructor(s)	Yes
Sub Class(es) Name(s)	MyViewHolder
Sub Class(es) Constructor(s)	Yes
Function(s) to be described	onCreateViewHolder onBindViewHolder onClick getItemCount
Variables	List<Meeting> meetingList Context context

This class extends RecyclerView.Adapter. A link will be provided in the Important Information section of this document for more information on the abstract class, RecyclerView, in the Android Studio documentation.

The constructor for this class contains meetingList and context

Sub Class Breakdown: MyViewHolder

Sub Class Name	MyViewHolder
Sub Class Constructor	Yes
Variables	TextView date TextView time TextView name

This class contains 3 variables (date, time and name, all of type TextView) which are initialised in the MyViewHolder constructor by a parameter view.

Function Breakdown: onCreateViewHolder, onBindViewHolder, onClick

Function Name	Return Type	Parameters	Variables
onCreateViewHolder	MyViewHolder	ViewGroup parent int viewType	(View) itemView
onBindViewHolder	Void	MyViewHolder holder int position	(Meeting) meeting (SimpleDateFormat) dateFormat (SimpleDateFormat) timeFormat
onClick	Void	View v	(Intent) intent

Both the onCreateViewHolder and onBindViewHolder methods are defined in the RecyclerView class (see Important Information). The onCreateViewHolder method is called when createViewHolder is used from RecyclerView and is used to create a new ViewHolder object with the parameters provided. The onBindViewHolder method is internally called when bindViewHolder is called from RecyclerView to update the ViewHolder contents with the item at the given position.

The onClick event in this scenario will create a new Intent object with the provided parameters and use it to create a new activity on the screen in the application. The MeetingInformation comes from MeetingInformation.java.

Class Breakdown: MeetingInformationActivity.java

Class Name	MeetingInformationActivity
Constructor(s)	None
Function(s) to be described	onCreate onClick
Variables	Meeting meeting View meetingView TextView date TextView time TextView clientName TextView meetingLocation MapFragment mapFragment

This class extends the AppCompatActivity abstract class for use. For more information on this class, see the Important Information section for a link to the part of the Android Studio documentation that concerns this abstract class.

Function Breakdown: onCreate, onClick

Function Name	Return Type	Parameters	Variables
onCreate	Void	Bundle savedInstanceState	(SimpleDateFormat) dateFormat (SimpleDateFormat) timeFormat (TextView) date (TextView) time (Button) notesButton
onClick	Void	View v	(Intent) intent

The onCreate event in this instance will set up the date and time for each client in a meeting. It will also create a new button on the screen called notesButton which has an onClick listener that will create a new Intent object to set up a new activity on the screen to provide meeting notes information.

Class Breakdown: MeetingNotes.java

Class Name	MeetingNotes
Constructor(s)	None
Function(s) to be described	onCreate onClick onResume onResponse onFailure onItemClick populateNotes
Variables	Meeting meeting SharedPreferences sharedPreferences RecyclerView rv RecyclerView.Adapter adapter RecyclerView.LayoutManager lm Button addNoteButton View meetingView TextView date TextView time TextView clientName ProgressBar progressBar

Function Breakdown: onCreate, onClick, onResume, populateNotes, onResponse, onFailure, onItemClick

Function Name	Return Type	Parameters	Variables
onCreate	Void	Bundle savedInstanceState	(SimpleDateFormat) dateFormat (SimpleDateFormat) timeFormat
onClick	Void	View v	None
onResume	Void	None	None

populateNotes	Void	None	None
onResponse	Void	Call<List<Note>> call Response<List<Note>> response	(ApiProtected) apiProtected
onItemClick	Void	Note note View view	(Intent) intent

The onCreate function in this class will set up the screen as usual with the addition of the addNoteButton which will be responsible for launching the screen for adding a new note via the onClick event that is attached to the button.

The populateNotes function will make sure that the correct notes are displayed for the meeting selected. If the method is successful in retrieving the required notes from the server then the onResponse method will trigger and successfully show the notes on the screen. Otherwise the onFailure event will trigger and subsequently throw an error.

The onResume event will occur when a ++screen is left and then returned to, to ensure that the information that is displayed on the screen is up to date when it has been changed by leaving the page.

Class Breakdown: Note.java

Class Name	Note
Constructor(s)	None
Function(s) to be described	Various Getters and Setters createFromParcel newArray describeContents writeToParcel
Variables	Integer id String heading String contents Creator<Note> CREATOR

This class is responsible for the creation of the Notes that are used throughout the application for describing meetings and keeping track of certain important information about clients. The class implements the Parcelable interface and therefore a link to the Android Studio documentation has been provided if more information is required. See the Important Information section for this link.

The constructor for this class is responsible for the creation of each note by taking information from the Parcel, in. Each note contains the id, heading and contents.

The class contains many getters and setters that allow the application to get the relevant information used throughout the different classes and set the information where necessary.

Function Breakdown: createFromParcel, newArray, describeContents, writeToParcel

Function Name	Return Type	Parameters	Variables
createFromParcel	Note	Parcel in	None
newArray	Note []	int size	None

describeContents	int	None	None
writeToParcel	Void	Parcel dest int flags	None

The creation of the Note is handled through the CREATOR. The createFromParcel function will take the Parcel in as a parameter and use it to return a new Note. The newArray function will return a new Note array of the size that has been provided via its parameter. The describeContents method is required by Retrofit and in the terms of this application there is no other instance that is required other than for it to return 0. The writeToParcel function is responsible for adding the information to the Parcel, dest, which consists of id, heading and contents.

Class Breakdown: NoteAdapter.java

Class Name	NoteAdapter
Constructor(s)	Yes
Sub Class(es) Name(s)	ViewHolder
Sub Class(es) Constructor(s)	Yes
Function(s) to be described	onCreate
Variables	List<Note> noteList Context context Meeting meeting ClickListener clickListener

This class extends the RecyclerView abstract class and a link will be provided in the Important Information to the API documentation for this class.

The constructor for this class will initialise the noteList and clickListener variables which are contained within any new NoteAdapter object.

Sub Class Breakdown: MyViewHolder

Sub Class Name	ViewHolder
Sub Class Constructor	Yes
Variables	TextView title TextView noteText

This class contains 2 variables (title and noteText of type TextView) which are initialised in the ViewHolder constructor by a parameter view. This sub class initialises a new onClick listener.

Function Breakdown: onCreateViewHolder, onBindViewHolder, onClick, getItemCount

Function Name	Return Type	Parameters	Variables
onCreateViewHolder	ViewHolder	ViewGroup parent int viewType	(View) itemView
onBindViewHolder	Void	ViewHolder holder int position	(Note) note
getItemCount	int	None	None

Both the onCreateViewHolder and onBindViewHolder methods are defined in the RecyclerView class (see Important Information). The onCreateViewHolder method is called when createViewHolder is used from RecyclerView and is used to create a new ViewHolder object with the parameters provided. The onBindViewHolder method is internally called when bindViewHolder is called from RecyclerView to update the ViewHolder contents with the item at the given position. The getItemCount function will return the size of the noteList.++

Class Breakdown: Order.java

Interface Name	Order
Constructor(s)	Yes
Function(s) to be described	Various Getters and Setters createFromParcel newArray writeToParcel describeContents
Variables	String orderNo Date dateOfOrder Parcelable CREATOR

This class implements the Parcelable interface. See the Important Information section for a link to the documentation concerning the Parcelable interface.

This class is responsible for the creation of new Order objects from the information taken from Parcels, using the class's constructor. There are getters and setters that are provided in the class that allow the application to retrieve an order's information and change it for the function writeToParcel to create a Parcel with the relevant information.

Function Breakdown: createFromParcel, newArray, writeToParcel, describeContents

Function Name	Return Type	Parameters	Variables
createFromParcel	Object	Parcel in	None
newArray	Order []	int size	None
writeToParcel	Void	Parcel dest int flags	None
describeContents	int	None	None

The createFromParcel will create a new Order object from the information provided by the parameter, Parcel in. The newArray method contained within the CREATOR variable alongside createFromParcel will create a new array of type Order with the dimension provided by the parameter, int size.

The writeToParcel method will write the information provided by the class (orderNo and dateOfOrder) to a Parcel, dest. The describeContents method is required by Retrofit and in the terms of this application there is no other instance that is required other than for it to return 0.

A toString method has been provided by the class so that a string output can be produced from the orderNo and dateOfOrder.

Class Breakdown: RetrieveTweets.java

Class Name	RetrieveTweets
Constructor(s)	Yes
Function(s) to be described	doInBackground onProgressUpdate onPostExecute scaledImageDimen loadImageFromURL
Variables	Activity page Context context ArrayList<twitter4j.Status> tweetsList

This class uses the Twitter4J Library alongside the Twitter API to handle the social media element of the application. For more information on this API, links have been provided in the Important Information section of this document to the documentation and GitHub of Twitter4J.

Function Breakdown: doInBackground, onProgressUpdate, onPostExecute, scaledImageDimen, loadImageFromURL

The size of the text in this table has been reduced to fit the vast amount of information that is needed.

Function Name	Return Type	Parameters	Variables
doInBackground	ArrayList<twitter4j.Status>	String … searchText	(ArrayList<twitter4j.Status>) tweetsList (ConfigurationBuilder) cb (TwitterFactory) twitterFactory (Twitter) twitter (Query) query (QueryResult) queryResult (int) titleTextSize (int) bodyTextSize (TextView) userText (StringBuilder) sb (ImageView) profileImage (TextView) tweetText (LinearLayout.LaoutParams) txtlp (MediaEntity []) tweetImages (ImageView) inTextImage (int []) dimen (LinearLayout.LayoutParams) layoutParams (LinearLayout) userLayer (LinearLayout) textLayer (LinearLayout) finalLayout
onProgressUpdate	Void	LinearLayout … values	(LinearLayout) newsLayout
onPostExecute	Void	ArrayList<twitter4j.Status> result	None
scaledImageDimen	int []	int heightSize Drawable image	(float) scaleValue (int) h (int) w (int []) scaledDimen

loadImageFromURL	Drawable	String url	(URL) imageURL (InputStream) is (Drawable) image

This class is called from the ClientProfileActivity class. The class will log into Twitter using the keys that have been provided, which are generated on a dummy Twitter account using the Twitter Developer tools. A Twitter viewer has been created so that it searches through tweets that use the '@' function related to Kerridge Commercial Systems, however, it avoids retweets. There is more information on this in the documentation for this library and a link will be provided in the Important Information section of this document. The rest of the for loop in this doInBackground function loops through each tweet that has been found, which then loads the profile picture, the username, the display name, the tweet text and the tweet images. All this information is added to a LinearLayout that already exists on the XML page.

The function itself only loads 2 tweets at once as to avoid any problems with running the program. The class extends AsyncTask (which is where the methods doInBackground, onProgressUpdate, onPostExecute are inherited from). The scaledImageDimen function returns the size that an image should scale to based on a locked height size. The loadImageFromURL method takes a s string containing a URL and converts that image to a Drawable that can be used by android studio.

Class Breakdown: SalesData.java

Class Name	SalesData
Constructor(s)	None
Function(s) to be described	onCreate getFormattedValue Multiple onClick events getGraph getAlert
Variables	BarChart salesChart TextView dateDisplay Button btnPrevious Button btnNext Button btnBack SimpleDateFormat dateFormat Date thisDate Calendar cal List<TOrders> SalesData ArrayList<String> months

This class is responsible for creating a graph to display all the sales data on. It accomplishes this using a library called MPAndroidChart which is from a GitHub repository. A link to this GitHub will be included in the Important Information section of this document.

Function Breakdown: onCreate, getFormattedValue, onClick(multiple), getGraph, getAlert

Function Name	Return Type	Parameters	Variables
onCreate	Void	Bundle savedInstanceState	(XAxis) xAxis

getFormattedValue	String	float value AxiBase axis	None
btnPrevious onClick	Void	View v	None
btnNext onClick	Void	View v	None
btnBack onClick	Void	View v	None
getGraph	Void	None	(Calendar) calendar (BarDataSet) barDataSet (BarData) theData
getAlert	Void	String alert	(AlertDialog.Builder) builder

The onCreate function will initialise the screen for the sales data graph alongside all the variables that are required to view everything on screen. At the testing phase, test data was used to make sure that everything was working as it should. Each button on the screen has an associated onClick event that will function labelled by the button (in relevance to the graph).

The getGraph method is responsible for creating the graph for the data that has been provided. It is made up of a simple for loop that will create the structure for the graph based on the data's range and then fill the graph with the relevant data. The getAlert method will display a message if the user attempts to view a year that has no records.

Class Breakdown: ScheduleMeetingActivity.java

Class Name	ScheduleMeetingActivity
Constructor(s)	None
Function(s) to be described	onCreate
	Multiple onClick events
	onDateSet
	onTimeSet
Variables	SharedPreferences sharedPreferences
	Button dateButton
	Button startTimeButton
	Button endTimeButton
	Button scheduleButton
	Integer pickerSelected
	EditText location
	Integer startHour
	Integer startMinute
	Integer endHour
	Integer endMinute
	Integer day
	Integer month
	Integer year
	Int START_TIME_PICKER
	Int END_TIME_PICKER
	SimpleDateFormat dateFormat
	SimpleDateFormat timeFormat

This class extends AppCompatActivity and implements the DatePickerDialog and TimePickerDialog interfaces. See the Important Information section for a link to Android Studio documentation for the class and interfaces mentioned above.

This screen handles a lot of information. This information includes the date and time that the meetings are held on. To split up this information, multiple buttons are used to separate the elements.

Function Breakdown: onCreate, onClick(multiple), onDateSet, onTimeSet

Function Name	Return Type	Parameters	Variables
onCreate	Void	Bundle savedInstanceState	(TextInputLayout) locationText (Button) scheduleButton (ImageButton) editClient
dateButton onClick	Void	View v	(DialogFragment) datePicker
timeButton onClick	Void	View v	(DialogFragment) timePicker
scheduleButton onClick	Void	View v	(Boolean) fail (Button) dateButton (String) date (Button) timeButton (String) time (String) location (TextView) clientName (String) clientNameString (TextView) clientLoc (String) clientLocString
onDateSet	Void	DatePicker view int year int month int dayOfMonth	(Calendar) c (String) dateString (Button) button
onTimeSet	Void	TimePicker view int hourOfDay int minute	(Button) button

The onCreate function in this class handles the initialisation of the buttons that are used on the screen. These buttons are dateButton, which will display a date picker when its onClick event is triggered, a timeButton, which will display a clock allowing the user to pick what time the meeting is at when its onClick event is triggered, and a scheduleButton, which will create the meeting when it is clicked.

When the date is selected from the calendar that is displayed when clicking dateButton, the information will be taken from the calendar and displayed in a string format as described by the onDateSet function. This is the same for onTimeSet, except this time the time will be taken from the clock display from the timeButton and displayed in its own string in the format hourOfDay, followed by a colon, then minute.

Class Breakdown: ServiceGenerator.java

Class Name	ServiceGenerator
Constructor(s)	None

Function(s) to be described	onCreate
Variables	OkHttpClient.Builder httpClient Retrofit.Builder builder

This class uses the Retrofit library to handle token authentication. See the Important Information section for a link to the documentation related to the Retrofit library.

It should also be noted that this class was built following a tutorial on creating a service generator and the link to this tutorial will also be provided in the Important Information section.

There are a lot of variables included in this class and that means a lot of information is handled that needs to be broken down. The dateFormat and timeFormat variables allow for the setting of the format for displaying the date and time on the screen. So instead of the default format that displays all the information for the date, dateFormat sets it so that only the day and month is displayed. The purpose of the start and end times are for the calculation of the meeting duration.

Function Breakdown: onCreate, onClick(multiple), showTimePicker, submitMeeting, onResponse, onFailure, onDateSet, onTimeSet

Function Name	Return Type	Parameters	Variables
onCreate	Void	Bundle savedInstanceBundle	(Calendar) c
dateButton onClick	Void	View v	None
startTimeButton onClick	Void	View v	None
endTimeButton onClick	Void	View v	None
scheduleButton onClick	Void	View v	None
showTimePicker	Void	None	(Calendar) c (Integer) hour (Integer) minute (TimePickerDialog) timePicker
showTimePicker	Void	Integer hour Integer minute	(TimePickerDialog) timePicker
showDatePicker	Void	None	(Calendar) c (Integer) year (Integer) month (Integer) day (DatePickerDialog) timePicker
submitMeeting	Void	None	(String) locationText (Calendar) c (Date) startDate (Date) endDate (CreateMeetingBody) createMeetingBody (ApiProtected) apiProtected (Call<Void>) call
onResponse	Void	Call<Void> call Response<Void> response	None

onFailure	Void	Call<Void> call Throwable t	None
onDateSet	Void	DatePicker view Int year Int month Int dayOfMonth	(Calendar) c
onTimeSet	Void	TimePicker view Int hourOfDay Int minute	(Calendar) c

The onCreate will initialise the activity as well as the buttons that are interactable on this screen. Each of these buttons have their own onClick event. In the case of the dateButton, startTimeButton and endTimeButton, clicking on them will bring up a simple interface to allow the user to pick either a date or time for each field on the screen. There are two showTimePicker functions, one that will work when a time has not been picked and therefore will choose the current date and time (default action) and theother that will take the hour and minute that has already been inputted into the two boxes, so that the user doesn't need to scroll around again.

The submitMeeting is the final step in this class. It will take all the information on the screen and schedule the meeting, saving it to the database. This is if none of the fields are invalid of null.

Class Breakdown: SplashScreenActivity.java

Class Name	SplashScreenActivity
Constructor(s)	None
Function(s) to be described	onCreate splashDelay run routeToPage
Variables	SharedPreferences sharedPreferences Long SPLASH_DURATION

This class extends the AppCompatActivity abstract class. A link will be provided in the important information section for more information.

Function Breakdown: onCreate, splashDelay, run, routeToPage

Function Name	Return Type	Parameters	Variables
onCreate	Void	Bundle savedInstanceState	None
splashDelay	Void	None	(Handler) handler
run	Void	None	None
routeToPage	Void	None	(Retrofit) Retrofit (ApiUnprotected) apiUnprotected

The onCreate event will call the function, splashDelay, which will display the splash screen. The splashDelay function contains a handler which will call run which in turn will call the function,

routeToPage, after a delay of *2500L* (which means that the splash will be shown for 2.5 seconds). The routeToPage function will load the LoginScreen class and move to the next screen.

Class Breakdown: TOrders.java

Class Name	TOrders
Constructor(s)	Yes
Function(s) to be described	onCreateDialog
Variables	None

TOrders is a simple class that contains two variables, quantity and date, and uses them to create a new TOrders object using a constructor. There are also two getter methods that allow the application to retrieve this information from a TOrders object.

Class Breakdown: UpcomingMeetings.java

Class Name	UpcomingMeetings
Constructor(s)	None
Function(s) to be described	onCreate onClick (Multiple onClick Events) populateMeetings onResponse onFailure
Variables	SharedPreferences sharedPreferences RecyclerView rv RecyclerView.Adapter adapter RecyclerView.LayoutManager lm Button viewAllButton Button scheduleButton Button clientListButton ProgressBar progressBar

This class will enable users to view the meetings that have been scheduled using the application.

Function Breakdown: onCreate, onClick(multiple), populateMeetings, onResponse, onFailure

Function Name	Return Type	Parameters	Variables
onCreate	Void	Bundle savedInstanceState	None
viewAllButton onClick	Void	View v	None
scheduleButton onClick	Void	View v	None
clientListButton	Void	View v	None
populateMeetings	Void	None	(ApiProtected) apiProtected (Call<List<Meeting>>) call
onResponse	Void	Call<List<Meeting>> call Response<List<Meeting>> response	(List<Meeting>) ml
onFailure	Void	Call<List<Meeting>> call Throwable t	None

The onCreate function will initialise all the buttons that are being used for this screen. Each button has an event listener and an onClick event attached to it so that it can function properly for the user when they need to use them. For the viewAllButton, the event will trigger an activity that will show every meeting that is happening. The scheduleButton will take the user to the scheduling screen which is managed by the ScheduleMeeting class. The clientListButton will show a screen containing all the clients that are currently stored on the application. An onResume function is used to call the populateMeetings function which is used to fill all the meetings in to the screen from the database whenever the user returns to this screen. Should the information be gathered correctly, the onResponse function will trigger and successfully fill the meetings. Otherwise the onFailure event will trigger throwing a suitable error for the situation.

Class Breakdown: UpdateMeetingBody.java

Class Name	UpdateMeetingBody
Constructor(s)	Yes
Function(s) to be described	None
Variables	Integer meetingId

This class has a meetingId that allows the application to know which meeting is being changed, along with the information that will be changed that is passed to the constructor (in which the object that is created contains the necessary information for the updated meeting including startDate, endDate, client, location and meetingId).

Class Breakdown: UpdateNoteActivity.java

Class Name	UpdateNoteActivity
Constructor(s)	None
Function(s) to be described	onCreate
	onClick
	updateNote
	onResponse
	onFailure
Variables	Note note
	EditText updateNoteBody
	EditText updateNoteHeading
	SharePreferences sharedPreferences
	Meeting meeting
	TextView updateNoteTitle

This screen is used when the user clicks on any note that is on the application. The view that is provided will allow the user to edit any pre-existing note and then click a button to save the changes.

Function Breakdown: onCreate

Function Name	Return Type	Parameters	Variables
onCreate	Void	Bundle savedInstanceState	(EditText) updateNoteBody (EditText) updateNoteHeading (Button) updateNoteButton
updateNote	Void	None	(String) heading

			(String) contents (int) noteID (int) meetId (UpdateNoteDetails) updateNoteDetails (ApiProtected) apiProtected (Call<Note>) call

The onCreate function is responsible for displaying the screen correctly. It will show the parts of the note on the screen that corresponds to the note that the user has clicked. From here, the user can edit both the heading and the body of the note. The onCreate function initialises the button, updateNoteButton, which has an event listener that will trigger when pressed by the user. It will call updateNote which is responsible for taking the information in the text boxes (provided neither are empty) and reupload them to the database. This can only happen if onResponse is called (meaning that the retrieval of the necessary information was successful). Otherwise onFailure will be called with the appropriate error.

Class Breakdown: UpdateNoteBody.java

Class Name	UpdateNoteBody
Constructor(s)	Yes
Function(s) to be described	None
Variables	Integer noteId

This class is responsible for describing the contents of any note that is created. These contents, barring noteId, are passed to the constructor within this class. A toString method is provided to allow for a string format of all the contents of an UpdateNotebody object. The noteId allows the application to know which note is actually being changed.

Important Information

This section contains links to any of the classes or interfaces that have been used from the Android Studio libraries (or any outside sources). These links have been included as this document is purely reserved for descriptions on the code that the Programming team has written for the application – not anything that has been used from the API itself.

Information on AppCompatActivity class:
https://developer.android.com/reference/android/support/v7/app/AppCompatActivity

Link to the GitHub of SundeepK who provided the CompactCalendarView Library:
https://github.com/SundeepK/CompactCalendarView

Information on the DatePickerDialog interface:
https://developer.android.com/reference/android/app/DatePickerDialog

Information on the Parcelable interface:
https://developer.android.com/reference/android/os/Parcelable

Information on RecyclerView.Adapter abstract class:
https://developer.android.com/reference/android/support/v7/widget/RecyclerView.Adapter

Information on Retrofit import: https://square.github.io/retrofit/

Information on the TimePickerDialog interface:
https://developer.android.com/reference/android/app/TimePickerDialog

A link to the API page of Twitter4J: http://twitter4j.org/en/

A link to the GitHub page of Twitter4J: https://github.com/Twitter4J/Twitter4J

Code Reference

A tutorial was used to create the ServiceGenerator class and therefore credit goes to Marcus Pöhls for the tutorial:

Pöhls. M, December 2014, *Retrofit- Token Authentication on Android*, Accessed 24/04/2019, <Available at: https://futurestud.io/tutorials/retrofit-token-authentication-on-android>