Swagafied

IOS Application

User Guidelines

V 1.

System Requirement:

OS version: iOS 9+
Xcode Version: Xcode 8+
Swift Version: Swift 3+

Application Overview:

Swagafied iOS Application consists of 3 main sections. They are

1. Search Tool
2. User Profile Tool
3. Account Tool

Search Tools gives you the facility to search your favorite kicks over thousands of available kicks. You can perform color wise, brand wise, Model wise search on this page.

User profile tools show your swagafied Portfolio, Watch list section along with Global timeline page.

Account Details consists for few sub sections like Notification page, Edit user profile page, Edit Password page, Terms Of Use page, Privacy Policy page and Login/Log out Page.

Search Tool :

Search page lists you all the products currently available in the remote server. Application enables you to like, unlike, share, add to your watchlist, remove from your watchlist, add to your portfolio and remove from your portfolio function for each of the products. You can move to product detail page by clicking the product image of info icon.

Here you can also perform a search functionality like using product Brand, Product Model name, Product Color.

Filter functionality provides you to select and search over collection currently available. Like User can select Adidas and KD and hit “MAKE IT HAPPEN”, Application will show currently available Adidas and KD kicks.

Filter like wise button filters all the kicks that user has liked currently.

Function for receiving data from server:

```
func getProductLists(){

    productListApiObject.didCompleteRequest = { [weak self] success in

      if(success){

        if (self!.productListApiObject.pageNumber == 0){
          self!.processArrayAndDownloadImage(products: (self?.productListApiObject.products)!)
        }

        self!.productListApiObject.pageNumber =  self!.productListApiObject.pageNumber + 1
        self!.bufferProductArray = self!.bufferProductArray + (self!.productListApiObject.products)

        if self!.productListApiObject.products.count == 30{
          self!.getProductLists()
        }else{

          (UIApplication.shared.delegate as! AppDelegate).isDataFetched = true
          (UIApplication.shared.delegate as! AppDelegate).dataSource = self!.bufferProductArray

          DispatchQueue.main.async {
            MBProgressHUD.hide(for: self!.view, animated: true)
          }
        }

      }else{

      }
    }

    let _ = NetworkManager.sharedManager.request(apiObject: productListApiObject)

    DispatchQueue.main.async {
      if (self.productListApiObject.pageNumber == 0){
        let hud = MBProgressHUD.showAdded(to: self.view, animated: true)
        hud.label.numberOfLines = 3
        hud.label.text = "Hold up a sec\n while we pull up\n all of our dope kicks. "
      }
    }
}
```

Network class responsible: ProductListAPI.Swift
API Endpoint:
"http://webolation.com/sneakers/web_service/product_list "

Funtions are responsible for search and Filter:

```
@IBAction func searchButtonAction(_ sender: AnyObject) {...}

@IBAction func filterButtonAction(_ sender: UIButton) {...}
```

User Profile Tool:

Timeline:

Timeline Page shows you all recent activities that has been performed by all the Swagafied users. Activities can be liking a Sneaker, Posting a Review, Add Sneaker to portfolio, Adding a Sneaker to Watchlist. From Timeline page user can also like a sneaker, Move to Sneaker detail page, can add to has/her watchlist. Comment button along with sneakers enables user to post a comment for a particular timeline object.

View all comments shows all the comments currently available for that particular sneaker. User can post his/her comments in All comments section too.

The share button enables user to share the Kick over all currently available options.

Like button can perform like or unlike the sneaker depending upon the current like state. If user haven't liked it yet then he can perform Like functionality else User can perform dislike functionality.

Functions for receiving Timeline data:

```
func getTimelineData(){
    timelineAPI.didCompleteRequest = {
        [weak self] success in

        self!.processArrayAndDownloadImage(products: self!.timelineAPI.products)
    }

    let _ = NetworkManager.sharedManager.request(apiObject:timelineAPI)
    DispatchQueue.main.async {
        MBProgressHUD.showAdded(to: self.view, animated: true)
    }
}
```

Network class responsible: TimelineAPI.Swift
API Endpoint:
"http://webolation.com/sneakers/web_service/ timeline"

Function for liking the product:

```
func likeButtonToogleActionn(indexPath: IndexPath, button: UIButton) {

    let productObject = self.dataSource[indexPath.row]
    let productListCell : TimelineTableViewCell = self.timelineTable.cellForRow(at: indexPath as IndexPath)
        as! TimelineTableViewCell
    MBProgressHUD.showAdded(to: self.view, animated: true)

    self.likeProductAPIObject.didCompleteRequest = {...}

    self.unlikeProductAPIObject.didCompleteRequest = {...}

    if productListCell.isLiked {
        let _ = NetworkManager.sharedManager.request(apiObject:self.unlikeProductAPIObject)
    }else{
        let _ = NetworkManager.sharedManager.request(apiObject:self.likeProductAPIObject)
    }

}
```

Network class responsible: LikeProductAPI.Swift
API Endpoint:
"http://webolation.com/sneakers/web_service/ like_product"

Function for fetching details about the product:

```
func didSelectInfoButtonn(indexPath: IndexPath) {

    let productObject = self.dataSource[indexPath.row]
    let _ = getPoductDetail(productid: productObject.productId!, completion: {
        (success: Bool) in

        self.selectedIndex = indexPath
        self.showProductDetailPage()

    })

}
```

Network class responsible: ProductDetail.Swift
API Endpoint:
"http://webolation.com/sneakers/web_service/product_details"

Share button Function:

```
func didSelectShareButtonn(indexPath: IndexPath, button: UIButton) {

    let cell : TimelineTableViewCell  = timelineTable.cellForRow(at: indexPath) as! TimelineTableViewCell
    let productObject : TimelineProduct = dataSource[indexPath.row]

    let text = "Find Lit Kicks @Swagafied. Download the app"
    if let image = cell.cellImageView.image {

        let vc = UIActivityViewController(activityItems: [image,text], applicationActivities: [])
        self.present(vc, animated: true, completion: nil)

        vc.completionWithItemsHandler = {...}

    }else{
        // Image not found

    }

}
```

Comment Button Function:

```
func didSelectAllCommentsButtonn(indexPath: IndexPath) {

    let productObject = self.dataSource[indexPath.row]
    let _ = getPoductDetail(productid: productObject.productId!, completion: {
        (success: Bool) in

        let messageVC : MessagesViewController = self.loadViewControllerFromStoryboard(storyboard:
            Configuration.Storyboard.MainStoryBoard, viewControllerIndentifier: Configuration.StoryboardVC.
            MessagesViewController) as! MessagesViewController

        messageVC.showReview = false
        messageVC.dataModel = self.productDetailAPI.product
        let cell : TimelineTableViewCell  = self.timelineTable.cellForRow(at: indexPath) as!
            TimelineTableViewCell
        messageVC.dataModel?.productMainImage = cell.cellImageView.image
        self.navigationController?.pushViewController(messageVC, animated: true)

    })
}
```

Network class responsible: AllCommentList.Swift
API Endpoint:
"http://webolation.com/sneakers/web_service/comment_list"

Posting a comment:

```
func postCommentAction(sender:UIButton){ ... }
```

API Endpoint:
"http://webolation.com/sneakers/web_service/ post_comment"

Liking a comment:

```
func didSelectLikeButton(index:IndexPath) {

    if showReview{ }
    else{

        likeCommentAPI.commentID = messageList[index.row].commentID!
        likeCommentAPI.didCompleteRequest = { (success: Bool) in

            let cell : MessagesTableViewCell = self.messageListTableView.cellForRow(at: index) as!
                MessagesTableViewCell
            cell.tagButton.setImage(UIImage(named: "like-selected"), for: .normal)

            self.messageList[index.row].likes = Int(self.messageList[index.row].likes!) + 1
            cell.likeButton.setTitle(" \(self.messageList[index.row].likes!) Likes", for: .normal)
            self.messageList[index.row].liked = true
        }

        let _ = NetworkManager.sharedManager.request(apiObject: likeCommentAPI)

    }
}
```

API Endpoint:
"http://webolation.com/sneakers/web_service/ like_comment"

Report a comment:

```
// MARK: - MessagesTableViewCellDelegate
func didSelectReportButton(index:IndexPath) {

    DispatchQueue.main.async {
        MBProgressHUD.showAdded(to: self.view, animated: true)
    }

    if showReview{ }
    else{

        reportCommentAPI.commentID = messageList[index.row].commentID!
        reportCommentAPI.didCompleteRequest = { }

        let _ = NetworkManager.sharedManager.request(apiObject: reportCommentAPI)

    }
}
```

API Endpoint:
"http://webolation.com/sneakers/web_service/ report_comment"

Portfolio:

Portfolio Page shows all the product list that user has added to his/her portfolio. Clicking on each product leads to portfolio details page. Here user can modify the size and price of the sneaker. Along with modifying user can share, add to watchlist, like the sneaker.

Fetching portfolio product list:

```
func getProductLists(){

    if let userID = userIDForPortfolio{
        portfoliotListApiObject.dafaultUserID = userID
    }

    portfoliotListApiObject.didCompleteRequest = {...}

    let _ = NetworkManager.sharedManager.request(apiObject:portfoliotListApiObject)

    DispatchQueue.main.async {
        MBProgressHUD.showAdded(to: self.view, animated: true)
    }
}
```

API Endpoint:
"http://webolation.com/sneakers/web_service/ report_comment"

Fetching details for a particular product:

```
func getPoductDetail(){

    productDetailAPI.productID = dataModel?.productId

    productDetailAPI.didCompleteRequest = { [weak self] success in

        self!.productCategoryLabel.text = self!.productDetailAPI.product!.catagoryName ?? ""
        self!.productTitleLabel.text = self!.productDetailAPI.product!.productName ?? ""
    }

    let _ = NetworkManager.sharedManager.request(apiObject:productDetailAPI)
}
```

API Endpoint:
"http://webolation.com/sneakers/web_service/ product_details"

Adding/ Removing Product from User Portfolio:

```
@IBAction func portfolioButtonAction(_ sender: AnyObject) {

    if let productInfo = DatabaseHelper.sharedHelper.fetchItemForID(id: (dataModel?.productId)!){

        if (productInfo.isAddedToPortfolio?.boolValue)! == true{

            let alert = UIAlertController(title: "Portfolio Update", message: "Are you sure you want to remove
                this item from your portfolio?", preferredStyle: UIAlertControllerStyle.alert)

            alert.addAction(UIAlertAction(title: "Ok", style: UIAlertActionStyle.default, handler: {
                action in
                MBProgressHUD.showAdded(to: self.view, animated: true)

                self.portfolioDelete.portfolioID = (self.dataModel?.productId)!
                self.portfolioDelete.didCompleteRequest = {...}

                let _ = NetworkManager.sharedManager.request(apiObject:self.portfolioDelete)

            }))
            alert.addAction(UIAlertAction(title: "Cancel", style: UIAlertActionStyle.cancel, handler: nil))

            self.present(alert, animated: true, completion: nil)
        }
        else{

            self.portfolioAdd.productID = (dataModel?.productId)!
            self.portfolioAdd.productSize = (dataModel?.size)!
            self.portfolioAdd.productPrice = (dataModel?.price)!
            self.portfolioAdd.didCompleteRequest = {...}

            let _ = NetworkManager.sharedManager.request(apiObject:self.portfolioAdd)
        }

    }
}
```

API Endpoints:
"http://webolation.com/sneakers/web_service/ remove_portfolio"
"http://webolation.com/sneakers/web_service/ add_portfolio"

Updating Senaker portfolio data:

```
@IBAction func sneakerFolioButtonAction(_ sender: AnyObject) {

    if self.navigationBar.pageHeader == "Watchlist"{...}
    else if self.navigationBar.pageHeader == "Portfolio"{

        if let shoeSizeData = self.shoeSelectedData{

            updatePortfolioAPI.portfolioID = dataModel?.id
            updatePortfolioAPI.sizeID = shoeSizeData.0
            updatePortfolioAPI.price = shoeSizeData.2

            updatePortfolioAPI.didCompleteRequest = {...}

            let _ = NetworkManager.sharedManager.request(apiObject:updatePortfolioAPI)
        }
    }
}
```

API Endpoint:
"http://webolation.com/sneakers/web_service/edit_portfolio"

Watchlist:

Watchlist Page shows the entire product list that user has added to his/her watchlist. Clicking on each product leads to watchlist details page. Here user can modify the size and price of the sneaker. Along with modifying user can share, add to portfolio, like the sneaker.

Fetching watchlist product list:

API Endpoint:

```
func getProductLists(){

    if let userID = userIDForPortfolio{
        watchListApiObject.dafaultUserID = userID
    }

    watchListApiObject.didCompleteRequest = {...}
    let _ = NetworkManager.sharedManager.request(apiObject:watchListApiObject)

    DispatchQueue.main.async {
        MBProgressHUD.showAdded(to: self.view, animated: true)
    }
}
```

"http://webolation.com/sneakers/web_service/watchlist"

Fetching details for a particular product:

```
func getPoductDetail(){

    productDetailAPI.productID = dataModel?.productId

    productDetailAPI.didCompleteRequest = { [weak self] success in

        self!.productCategoryLabel.text = self!.productDetailAPI.product!.catagoryName ?? ""
        self!.productTitleLabel.text = self!.productDetailAPI.product!.productName ?? ""
    }

    let _ = NetworkManager.sharedManager.request(apiObject:productDetailAPI)
}
```

API Endpoint:
http://webolation.com/sneakers/web_service/ product_details

Adding/ Removing Product from User Watchlist:

```
@IBAction func watchListButtonAction(_ sender: AnyObject) {

    if let productInfo = DatabaseHelper.sharedHelper.fetchItemForID(id: (dataModel?.productId)!){

        if (productInfo.isAddedToWishList?.boolValue)! == true{

            let alert = UIAlertController(title: "Watchlist Update", message: "Are you sure you want to
                remove this item from your watchlist?", preferredStyle: UIAlertControllerStyle.alert)

            alert.addAction(UIAlertAction(title: "Ok", style: UIAlertActionStyle.default, handler: {
                action in
                MBProgressHUD.showAdded(to: self.view, animated: true)

                self.watchlistDelete.watchListID = (self.dataModel?.productId)!
                self.watchlistDelete.didCompleteRequest = {...}

                let _ = NetworkManager.sharedManager.request(apiObject:self.watchlistDelete)

            }))
            alert.addAction(UIAlertAction(title: "Cancel", style: UIAlertActionStyle.cancel, handler: nil))

            self.present(alert, animated: true, completion: nil)
        }
        else{

            self.watchlistAdd.productID = (dataModel?.productId)!
            self.watchlistAdd.productSize = (dataModel?.size) ?? "4"
            self.watchlistAdd.productPrice = (dataModel?.price) ?? "---"
            self.watchlistAdd.didCompleteRequest = {...}

            let _ = NetworkManager.sharedManager.request(apiObject:self.watchlistAdd)
        }

    }

}
```

API Endpoints:
"http://webolation.com/sneakers/web_service/remove_watchlist"
"http://webolation.com/sneakers/web_service/add_watchlist"

Updating Sneaker watchlist data:

```
@IBAction func sneakerFolioButtonAction(_ sender: AnyObject) {

    if self.navigationBar.pageHeader == "Watchlist"{

        if let shoeSizeData = self.shoeSelectedData{

            updateWatchlistAPI.watchlistID = dataModel?.id
            updateWatchlistAPI.sizeID = shoeSizeData.0
            updateWatchlistAPI.price = shoeSizeData.2

            updateWatchlistAPI.didCompleteRequest = {
                (success) -> () in

                if success{...}
            }

            DispatchQueue.main.asyncAfter(deadline: .now()+0.1){...}

            let _ = NetworkManager.sharedManager.request(apiObject:updateWatchlistAPI)
        }
    }
    else if self.navigationBar.pageHeader == "Portfolio"{...}
}
```

API Endpoint:
http://webolation.com/sneakers/web_service/edit_watchlist

Account Tools :

Edit Profile:

Edit profile page allows user to modify all user related information like Updating Name, User Name, Date of Birth, Company info, website address, Phone number and Address. Along with all this information user can modify his/her profile picture by simply tapping on the image.

Updating User Data:

```
override func didSelectRightButton(sender:UIButton) {

    DispatchQueue.main.async {
        MBProgressHUD.showAdded(to: self.view, animated: true)
    }

    updateUserProfile.paramList1 = ["user_id": Utility.getUserID() ,
                                    "name": "\(firstName.text ?? "")_\(lastName.text ?? "")",
                                    "user_type": "1",]
    updateUserProfile.paramList2 = ["date_of_birth":"\(dateOfBirth.text ?? "")","company":"\(company.
        text ?? "")","website":"\(website.text ?? "")"]
    updateUserProfile.paramList3 = ["phone":"\(phone.text ?? "")" ,"city" : "\(city.text ?? "")","state" : "\
        (state.text ?? "")"]
    updateUserProfile.paramList4 = ["zip" : "\(zip.text ?? "")","country" : "\(country.text ??
        "")" ,"username": "\(userName.text ?? "")"]

    updateUserProfile.didCompleteRequest = {...}

    let _ = NetworkManager.sharedManager.request(apiObject:self.updateUserProfile)
}
```

API Endpoint:
http://webolation.com/sneakers/web_service/update_user

Upload User Avatar Image:

```
if self?.imageChanged == true{

    self?.imageUploadAPI.image = self?.userImage
    self?.imageUploadAPI.uploadImage()
    self?.imageUploadAPI.didCompleteRequest = { [weak self] success in

        DispatchQueue.main.async {
            MBProgressHUD.hide(for: self!.view, animated: true)
        }

        if success == true{...}
        else{...}
    }

}
```

API Endpoint: http://webolation.com/sneakers/web_service/update_profile_picture

Fetching User saved Data:

```
func getUserData(){

    userDataAPI.didCompleteRequest = {
        (success) in
      self.loadData()
    }

    if imagePickerDisplayed == false{
        let _ = NetworkManager.sharedManager.request(apiObject: userDataAPI)
    }else{
        imagePickerDisplayed = true
    }

}
```

API Endpoint:
http://webolation.com/sneakers/web_service/edit_watchlist

Terms of Use & Privacy Policy:

This section of the Application informs you about the legal terms, how user should use the application.

Login & LogOut :

One of the most important feature of the application, this enables application to identify the user by its credentials. Logout removes all your personal data by simply wiping off the database.

Application Level configurations:

Updating one signal ID:

Update "oneSignalAppID" variable in configuration.Swift file.

Updating Google Ad Configuration:

Step 1: Update "googleAdAppID" variable in configuration.Swift file.
Step 2: Update "adUnitID" variable in configuration.Swift file.

Updating Base API URL:

Update "baseAPIUrlStating" variable in configuration.Swift file.

Updating API endpoints:

Modify "path" structure variable inside "API" structure in configuration.Swift file.

Integrate your own Facebook app:

1. update "FacebookAppID" in info.plist file.
2. Update Url schemes with your own app id.

Updating Onesignal, Facebook, Google Ad library:

Just run "Pod Update" command in Terminal. It will fetch all latest libraries for you.