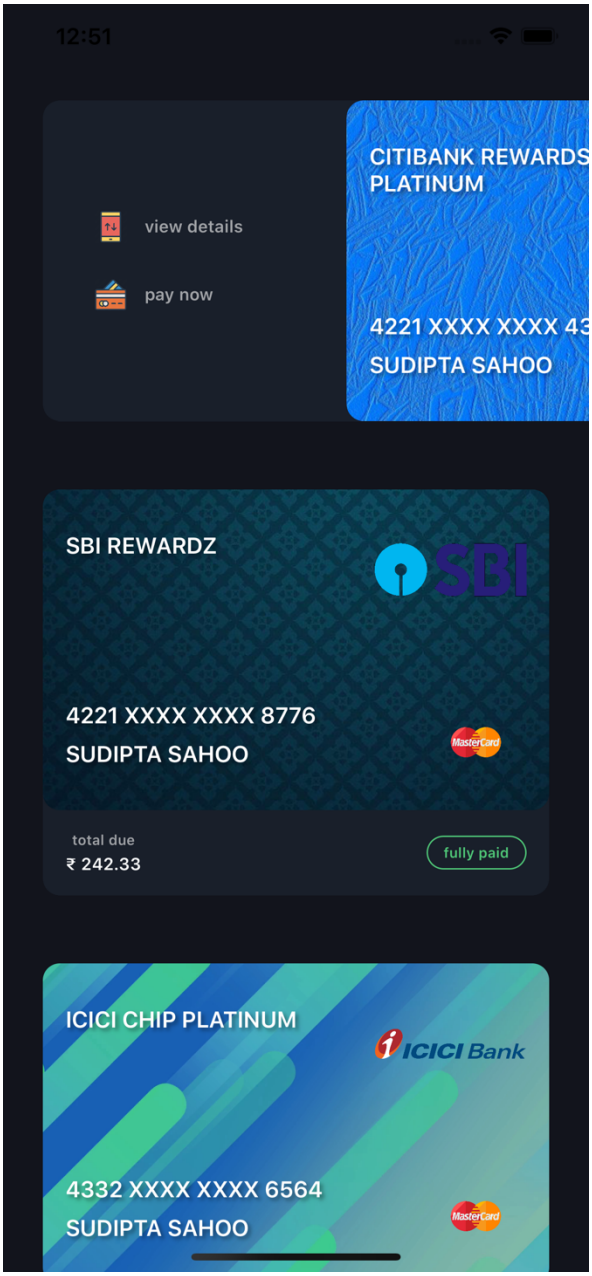
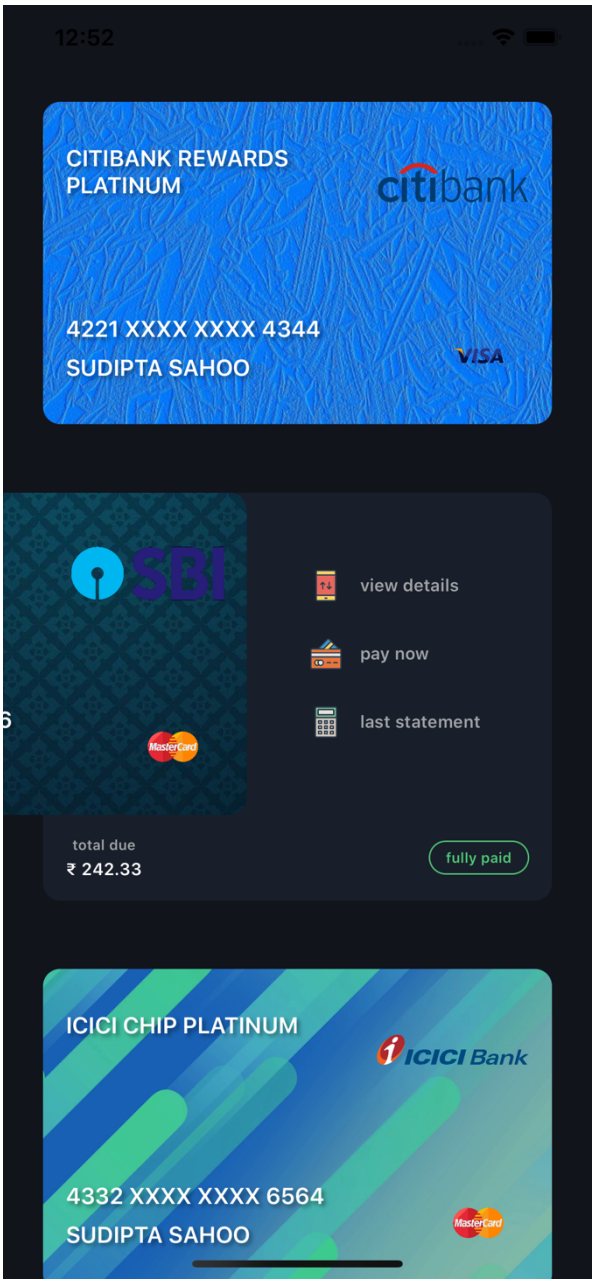
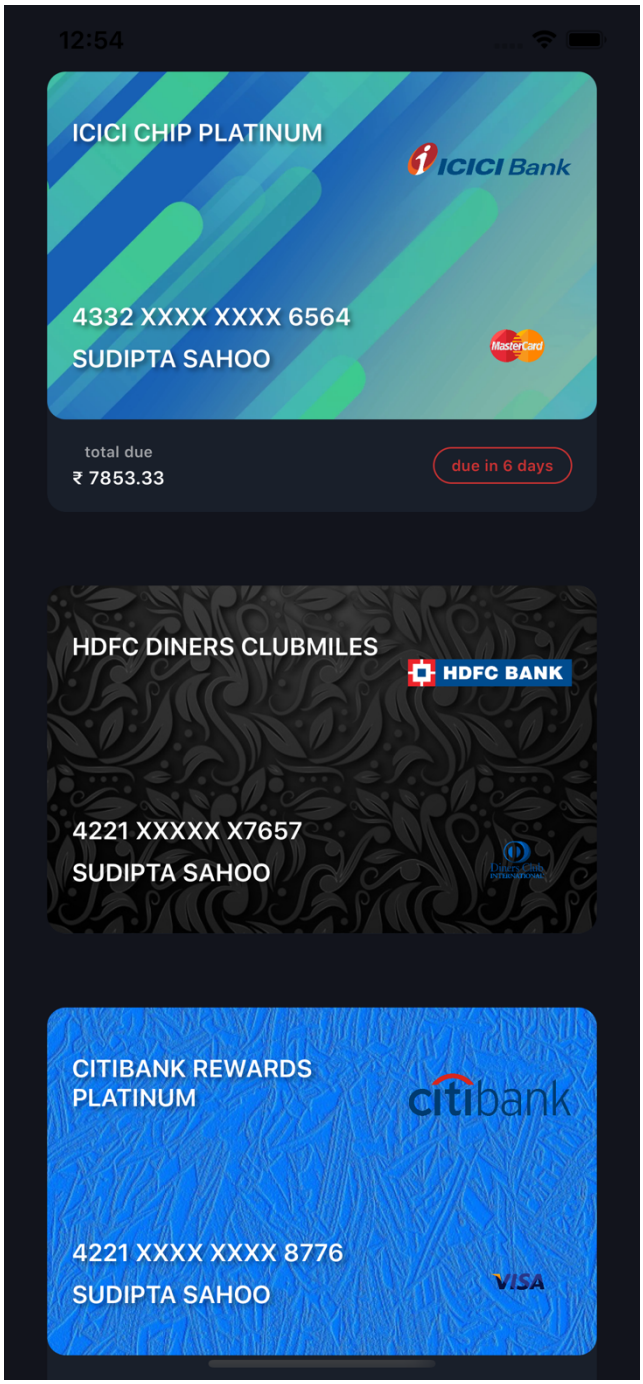
Swippy App

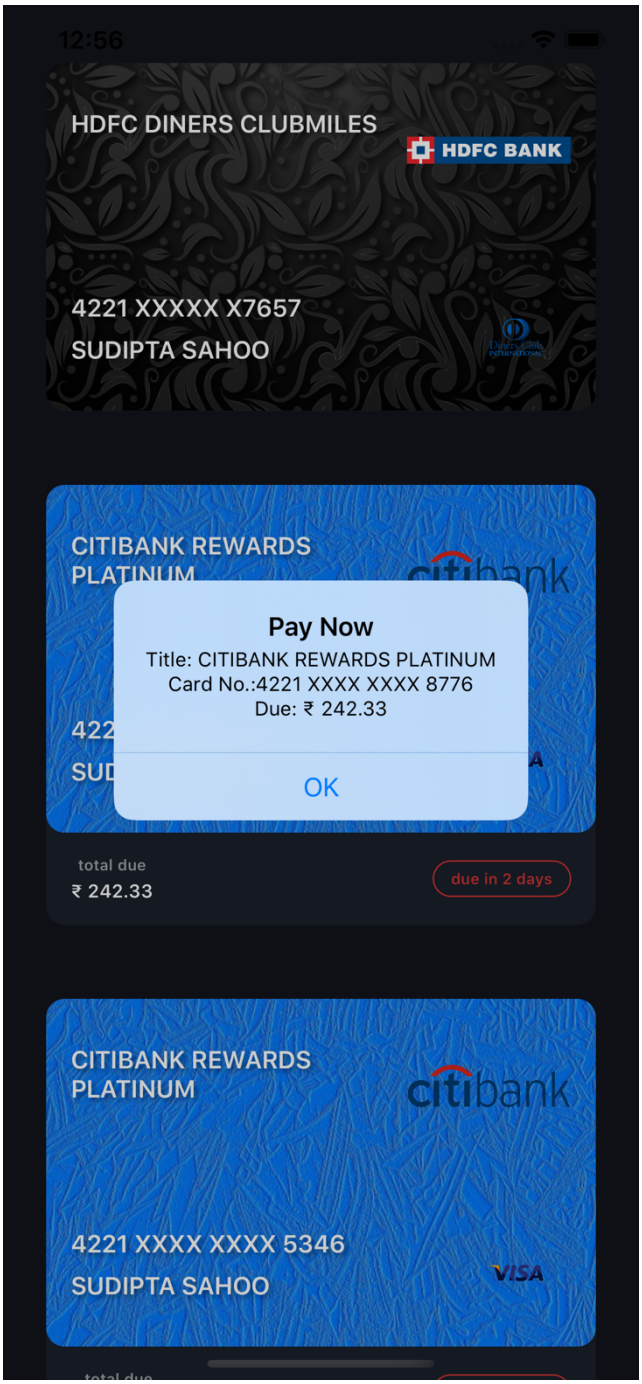
CRED iOS Assignment Submission by Sudipta Sahoo

Phone: 8080842428
Email: me@sudiptasahoo.in
Linkedin: https://www.linkedin.com/in/sahoosudipta/









Project Details:

- Project Name: **Swippy**
- Developed with ❤️ and dedication using Swift 5 and XCode 11.3

Project Instructions:

- **Github URL:** https://github.com/sudiptasahoo/Swippy-iOS
- Open ***Swippy.xcworkspace*** file
- If required please do ***pod install***
- Scheme to be used to run the project: **Swippy**

Project Dependencies:

- Dependencies included through pod:
  - "RxSwift"
- Programming Language*:* ***Swift 5***
- IDE*: XCode 11.3*

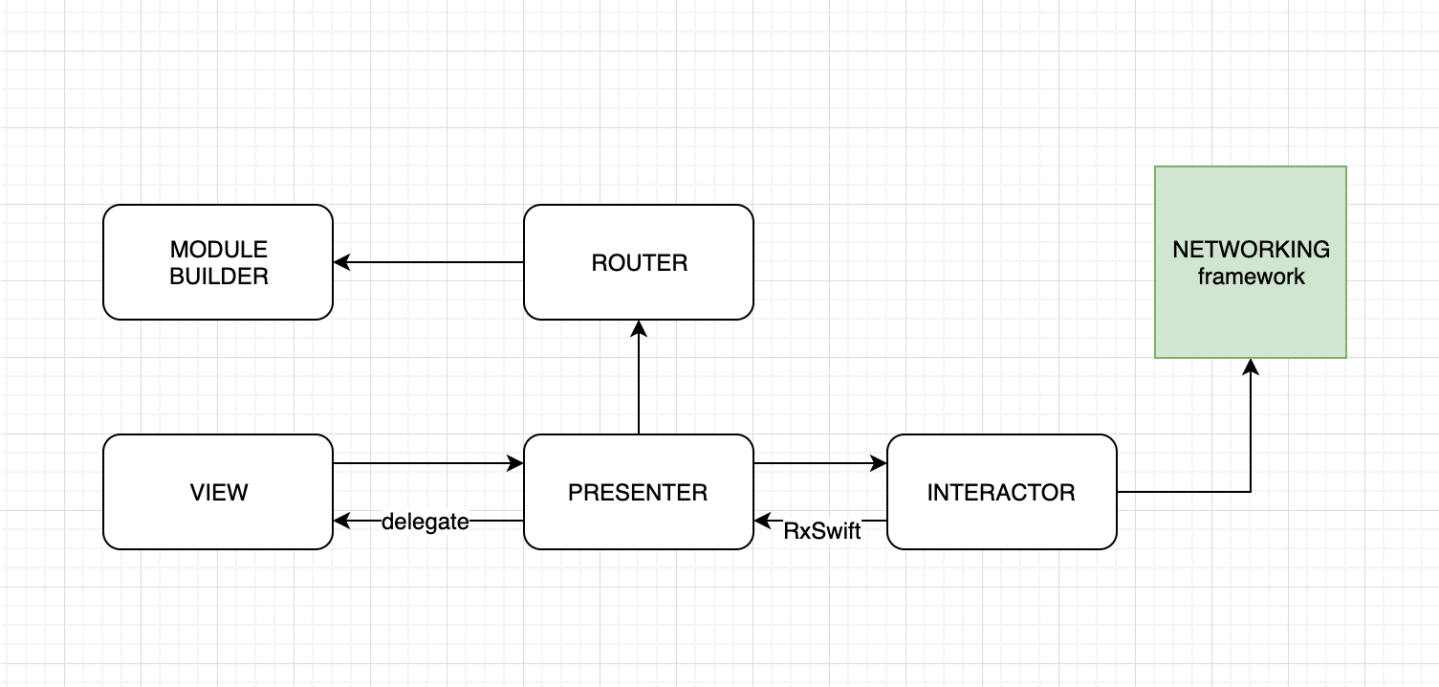
Project Structure:

- Architecture used: **VIPER**
- High Level Workspace Modules:
  - **Swippy**– The Main iOS Applicatio
  - **Source/Module –** Each individual module in the whole application. This project has just one module named **PaymentCards**
  - **Source/Module/PaymentCards –** It has the VIPER layers.

Project Sanitizations done:

- The following sanitizations have been run and the project has been found clean of such issues:
  - Main thread sanitizer

Project UML Diagram:



- **VIEW**: This is dumb and takes all data from the PRESENTER.
- **PRESENTER*:*** This layer always has the latest data and communicates with VIEW and ROUTER. Data with the PRESENTER is the only source of truth at any point in the execution time.
- **INTERACTOR:** This layer only has the business logic like form validations, getting data from networking layer and then modify it as required.
- **ROUTER:** This layer communicates with the BUILDER and navigates across the app.
- **BUILDER:** This layer creates modules resolving and injecting all the dependencies and returns the module VC. This can be modified to return the whole dependency container, if required in a bigger project.
- **ENTITY:** The PCardResponse model confirms to Decodable protocol making it easily understandable by other layers.
- **NETWORKING:** This is the framework for all HTTP tasks. This project implementes MockNetwork service here. But a real project will have more robust Networking layer here. Ref: https://github.com/sudiptasahoo/FlickrFeed/tree/master/EasyNetworking

Note:

- RxSwift Observable has been used only to bind Presenter and Interactor in this project. It could have been used to bind View and Presenter too.

Features:


```
        {
            "id": "48822393341",
            "title": "SBI REWARDZ",
            "number": "4221 XXXX XXXX 8776",
            "card_holder_name": "SUDIPTA SAHOO",
            "type": "MASTERC",
            "bank_name": "SBI",
            "due": {
                "id": "123243",
                "amount": 242.33,
                "currency": "₹",
                "state": "fully_paid"
            },
            "actions": ["view_details", "pay_now", "view_last_statement"]
        },
        {
            "id": "48822394455",
            "title": "ICICI CHIP PLATINUM",
            "number": "4332 XXXX XXXX 6564",
            "card_holder_name": "SUDIPTA SAHOO",
            "type": "MASTERC",
            "bank_name": "ICICI",
            "due": {
                "id": "128768",
                "amount": 7853.33,
                "currency": "₹",
                "state": "due in 6 days"
            },
            "actions": ["view_details", "pay_now", "view_last_statement"]
        },
```


- The Card listing is being rendered from the JSON in the **MockData.json** file.
    - ***due.state***:
        - ***fully_paid* –** If there are no dues.
        - ***<String>* -** String mentioning the days to due date. Eg. Due in 3 days
    - ***bank_name:***
        - Name of the card issuing bank.
    - ***type:***
        - The card network type. Eg. Visa
    - ***actions:***
        - This has list of pre-defined list of actions. Any new action introduced on the server side will be ignored in this project.
- **CardConfig.swift**:
    - `cardSwipeBehaviour, cardAutoClosingBehaviour`
        - The parameters here can be played with to alter the swipe experience.
    - `openPercentage, panThresholdPercentage, demoSwipePercentage`
        - These parameters can be altered to achieve a different swipe distance.

Scopes of improvement:

- Unit Test cases for Card Listing end to end
    - With overall code coverage: 75% +
- Code:
    - **CardWidth** and **CardHeight** has been calculated wrt. the device screen width for ease. In real project this can be taken wrt the UITableViewCell width.
    - As card width and height have been taken wrt the device screen width, the app will render bigger cards on larger screen devices like iPad.
    - Many Autolayout codes are verbose in the project. In real project, extensions could be created over UIView for cleaner code.
    - Pagination support is given in the presenter and interactor APIs but not implemented properly.
    - RxSwift is being used to bind Presenter and Interactor. RxSwift could be used to bind View and Presenter too.
    - ***ViewState*** provision has been provided in the Card listing screen but it is not implemented properly.
    - ***NetworkService*** will be way more robust with much more functionalities. It will be full-fledged code to do any kind of network operations.
        - Ref: https://github.com/sudiptasahoo/FlickrFeed/tree/master/EasyNetworking
    - Better exception handling.
    - Better documentation and detailed code documentation.
- I would love to discuss more with your team if given the opportunity.

Thank you

- Sudipta Sahoo
I can be reached at +91 8080842428 or me@sudiptasahoo.in