Group 09

VFund
Software Architecture Document

Version 1.0

Revision History

Date	Version	Description	Author
11/12/2020	1.0	Preliminary version	Vo Minh Lam
15/12/2020	1.1	Updates	Nguyen Thanh Minh Duy
19/12/2020	1.2	Class Diagrams	Nguyen Duy Thien Kim

Table of Contents

Software Architecture Document

1. Introduction

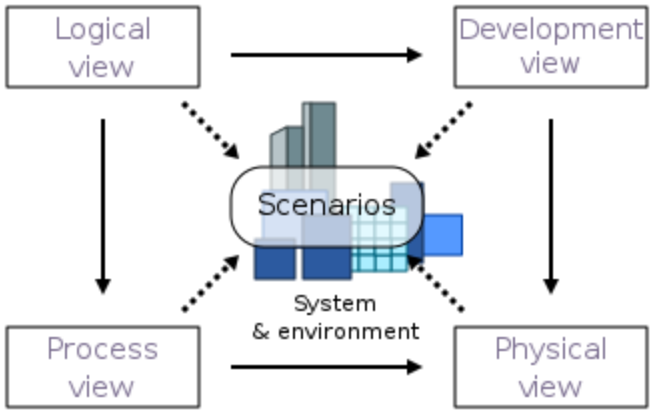
This VFund project creates the opportunities for people to join hands to help people who might need it. It organizes a much smaller scale of fundraising and the donation could be as little as people can afford. Fundraising can be authorized faster. Big events will have their own authorized mark to ensure that the fund project is legal and reliable.

This document elaborates the software architecture document for “VFund system”. The system architecture is abstracted into many views and components which are explained in detail. The document follows the 4+1 view models as the reference model for this document.

1.1 Purpose

The Software Architecture Document (SAD) provides a comprehensive architectural overview of the VFund application. It presents a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

In order to depict the software as accurately as possible, the structure of this document is based on the “4+1” model view of architecture. The “4+1” view model allows various stakeholders to find what they need in the software architecture.



1.2 Scope

The software architecture document applies to each static and dynamic aspect of the system. Since the 4+1 view model is used as the reference model, it corporates many views of the system, thus makes the document complete and consistent.

Under the static behavior of the system, the document discusses the class diagrams,

package diagrams and other static architecture designs. Dynamic aspects of the system are elaborated using case realizations and system sequence diagrams.

1.3 Definition, Acronyms and Abbreviations

OOP - Object Oriented Programming
SAD - Software Architecture Document
API - Application Programming Interface

1.4 References

- Architectural Blueprints—The "4+1" View Model of Software Architecture, Philippe Kruchten, November 1995: http://www3.software.ibm.com/ibmdl/pub/software/rational/web/whitepapers/2003/Pbk4p1.pdf

1.5 Overview

In order to fully document all the aspects of the architecture, the Software Architecture Document contains the following subsections.

Section 2: Describes the architectural goals and constraints of the system

Section 3: Describes the use-case model of application

Section 4: Describes the architecture with components and relationships among them

Section 5: Describes how the system is deployed by mapping the components in Section 4 to machine running them

Section 6: Describes the folder structures for the code for all components described in Section 4

2. Architectural Goals and Constraints

This section describes the software requirements and objectives that have some significant impact on the architecture.

1. Security

- All sensitive user data such as phone number, emails, ID or bank account are encrypted.
- A user can retrieve forgotten password by a link or code sent to their registered email.
- If a user fails to input the correct password more than 5 times, the system will automatically deactivate their account for 30 minutes.

2. Performance

- Client/ Server Connection's minimum bandwidth: 500KB/s.

- All screens and information must be ready-to-read on users' ends, provided Internet connection between server/client is normal

3. Usability

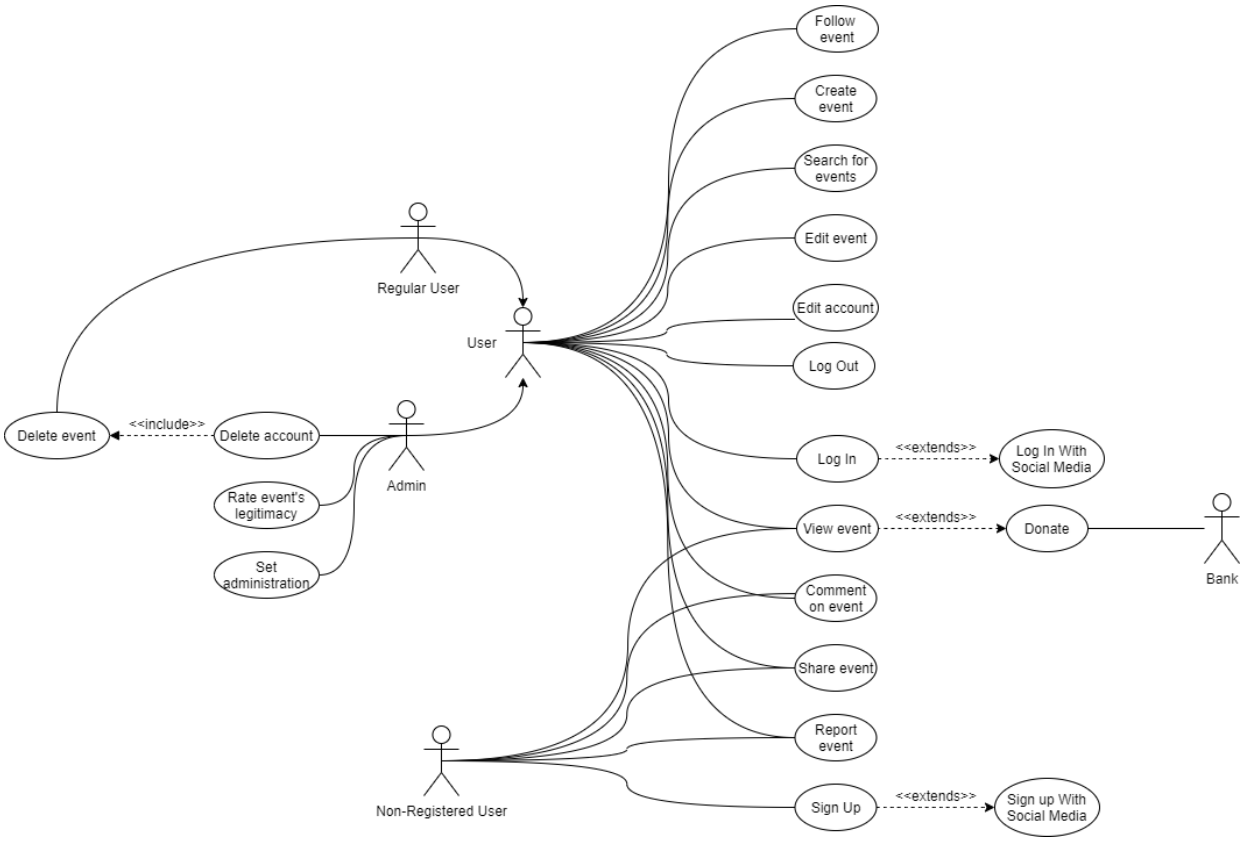
- Donate button is highlighted in Donation area of the screen for easier navigation

4. Technical Platform

- Platform: Android
- Version: 5.0 or above
- RAM: 1GB
- IDE: Android Studio
- Server Database: SQL Server

3. Use-Case Model

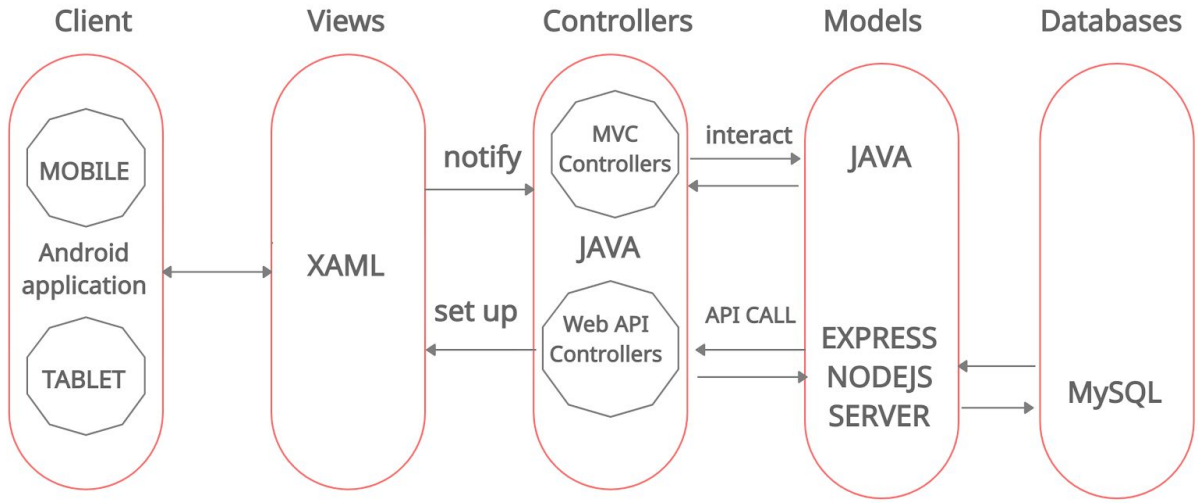
This section provides the Use-Case diagrams for VFund application. Each use-case specification has been described in Use-case Specification Document.



4. Logical View

Mobile application follow the construction of **client side rendering** and modeling to **MVC**.

This contain 3 big components:



- **Database**: using mySQL and SQLite, this component's duty is to store all the information about the user and the system (ex: donation event, user account,...) by linking tables.

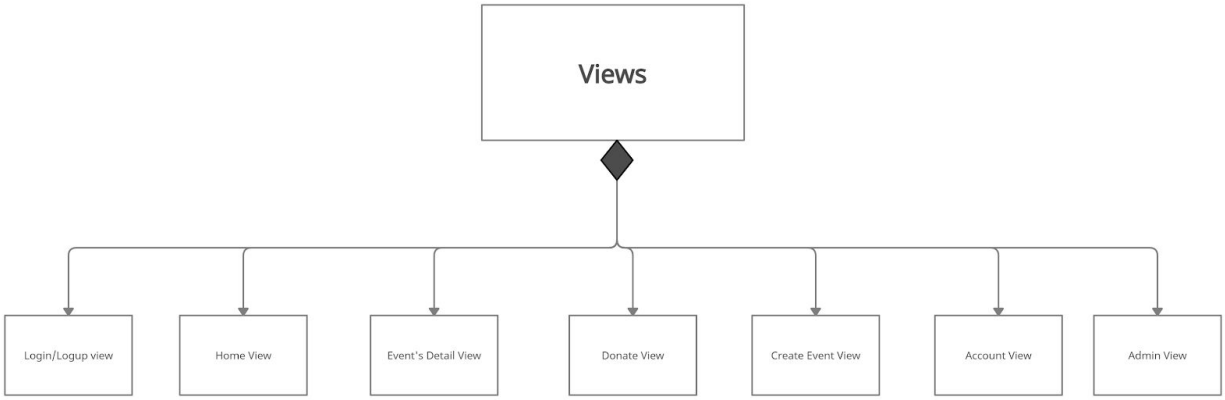
- **Models**: The application model will be built by java, handling all the inner interaction of user preferences. On the other hand, a server is built by js(express,node) to handle all api calls to make an event or donation,...

- **Controllers**: All the controllers will be built by java, handling all the user interaction with the application. There 2 types of Controllers, 1 are made to handle basic logic of the user inner interaction, the other are made to handle API call between server and application to create events or donations, etc

- **Views:** This component will render all the user interfaces, handling logic events made by the user (ex: clicks, swipe,...)

4.1 Component: Views

The main goal of this component is to render and update the user interface (UI) and interact with controllers after the user's interaction with the app.

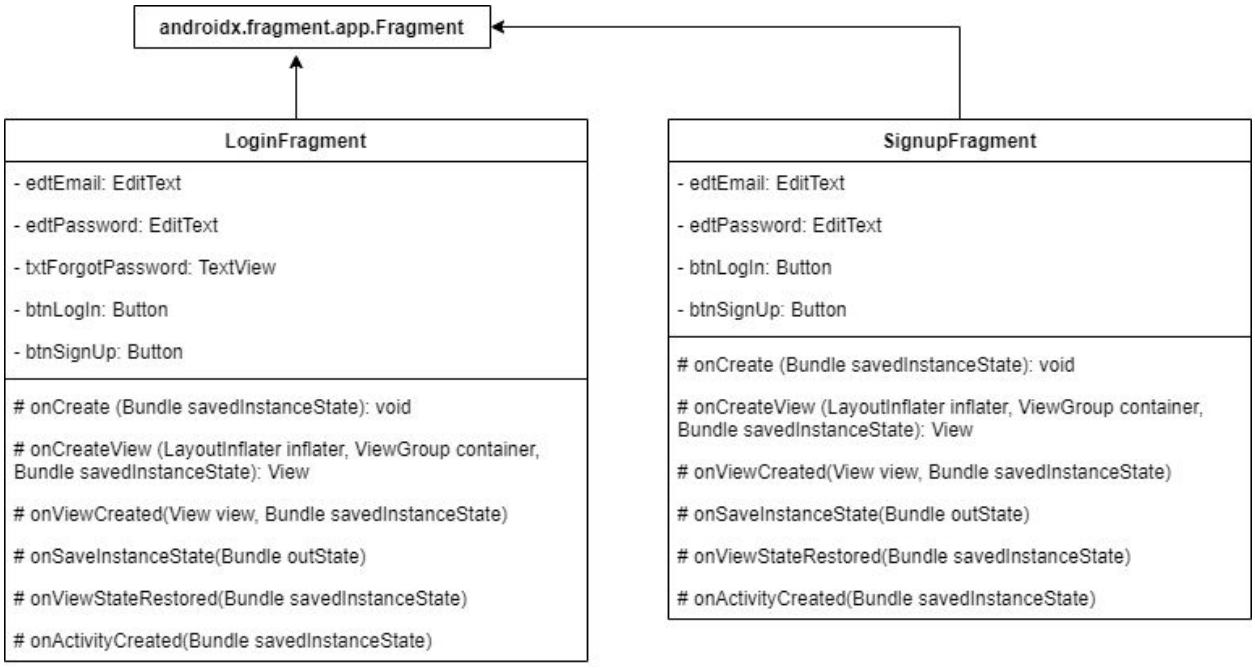


4.1.1 Login/Sign up view

- **Responsibilities**

  The main responsibility of this view is to display an interface for user to create an account or log in with an existed account

- **Class Diagram:**

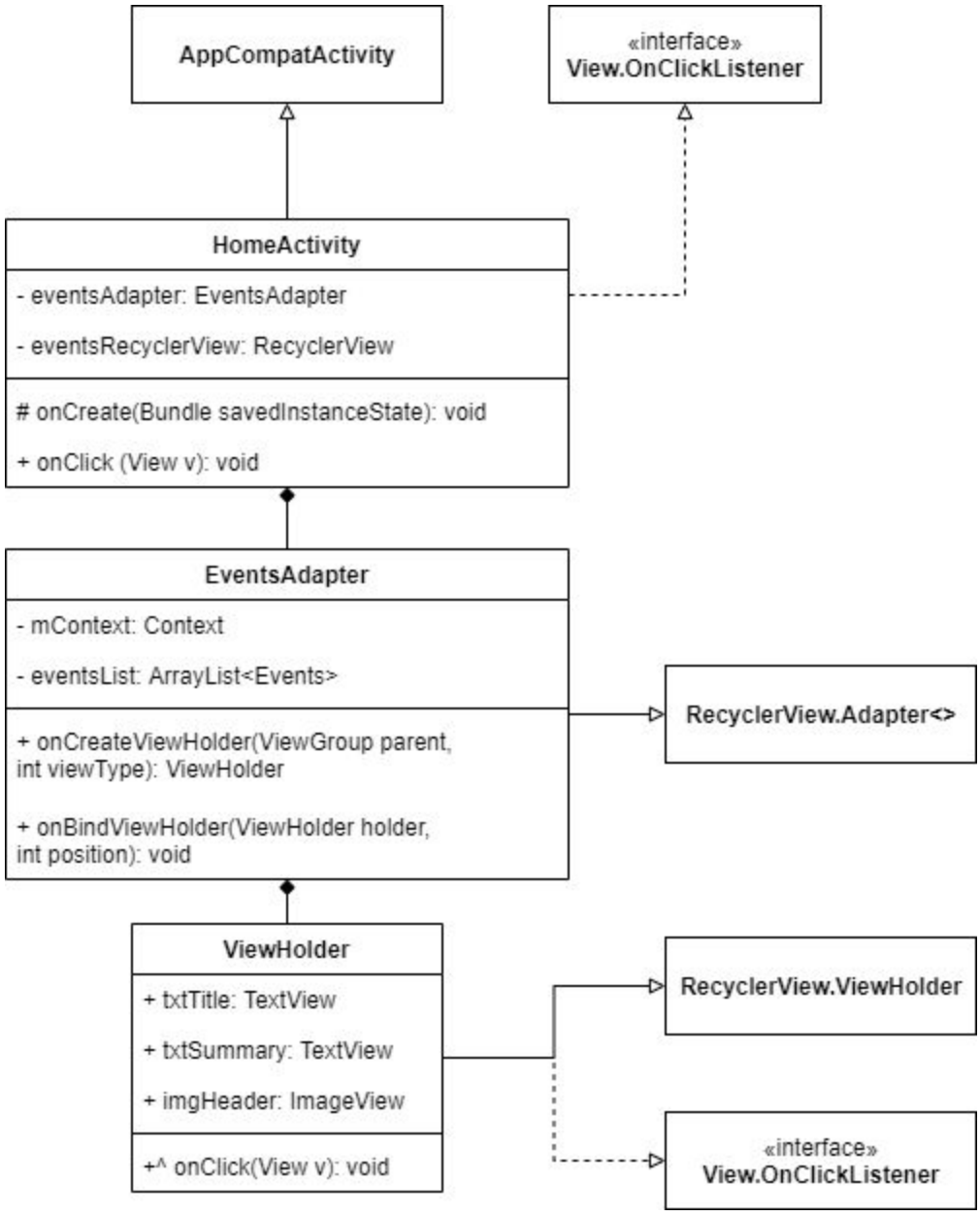


4.1.2 Home view

- **Responsibilities**

  The main responsibility of this view is to display a list of current events, user can also proceed to other views like account view, detail event view, create event view,...

- **Class Diagram:**

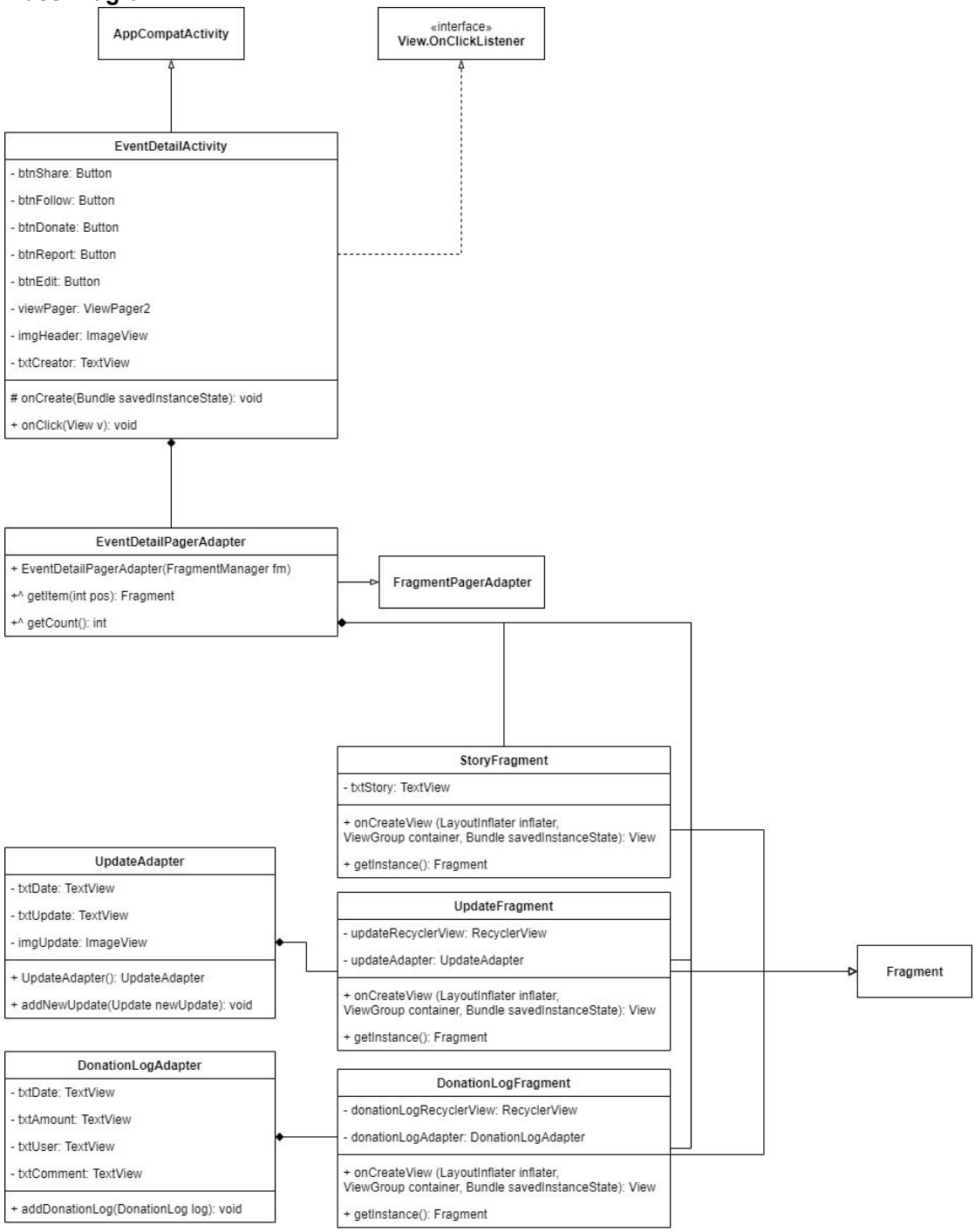


4.1.3 Event's Detail View

- **Responsibilities**

  The main responsibility of this view is to display every information about the selected event. User can also hit the donate button to proceed to donate view and share the event with the share button.

- **Class Diagram:**

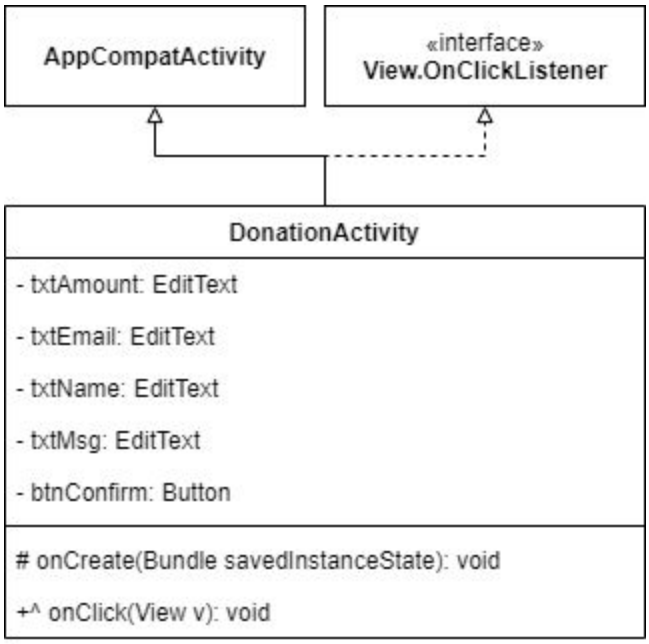


4.1.4 Donate View

- **Responsibilities**

  The main responsibility of this view is to display fields for users to fill in the information and make a donation to the event's holder.

- **Class Diagram:**

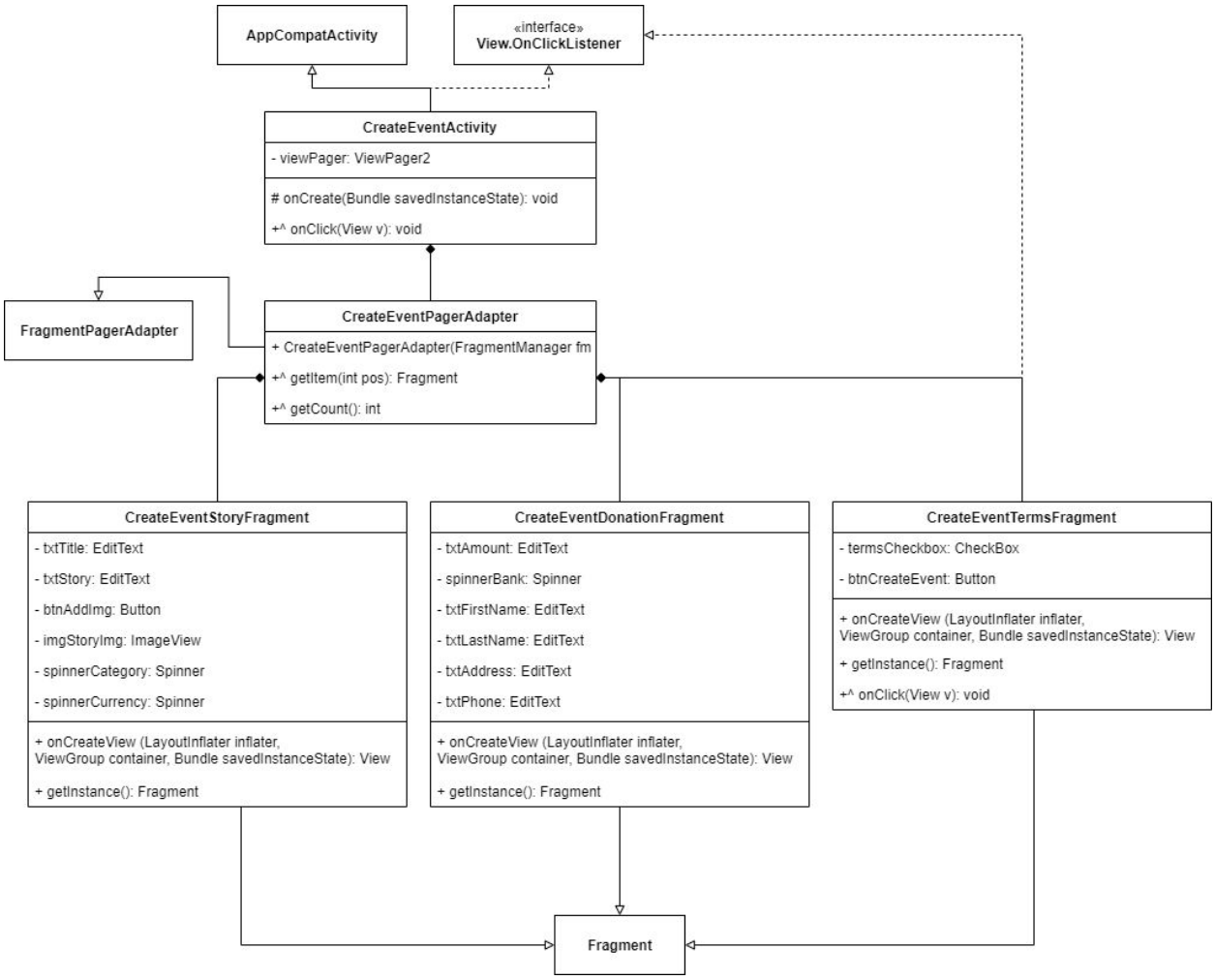


4.1.5 Create Event View

- **Responsibilities**

  The main responsibility of this view is to display fields for users to fill in the information to create an event.

- **Class Diagram:**



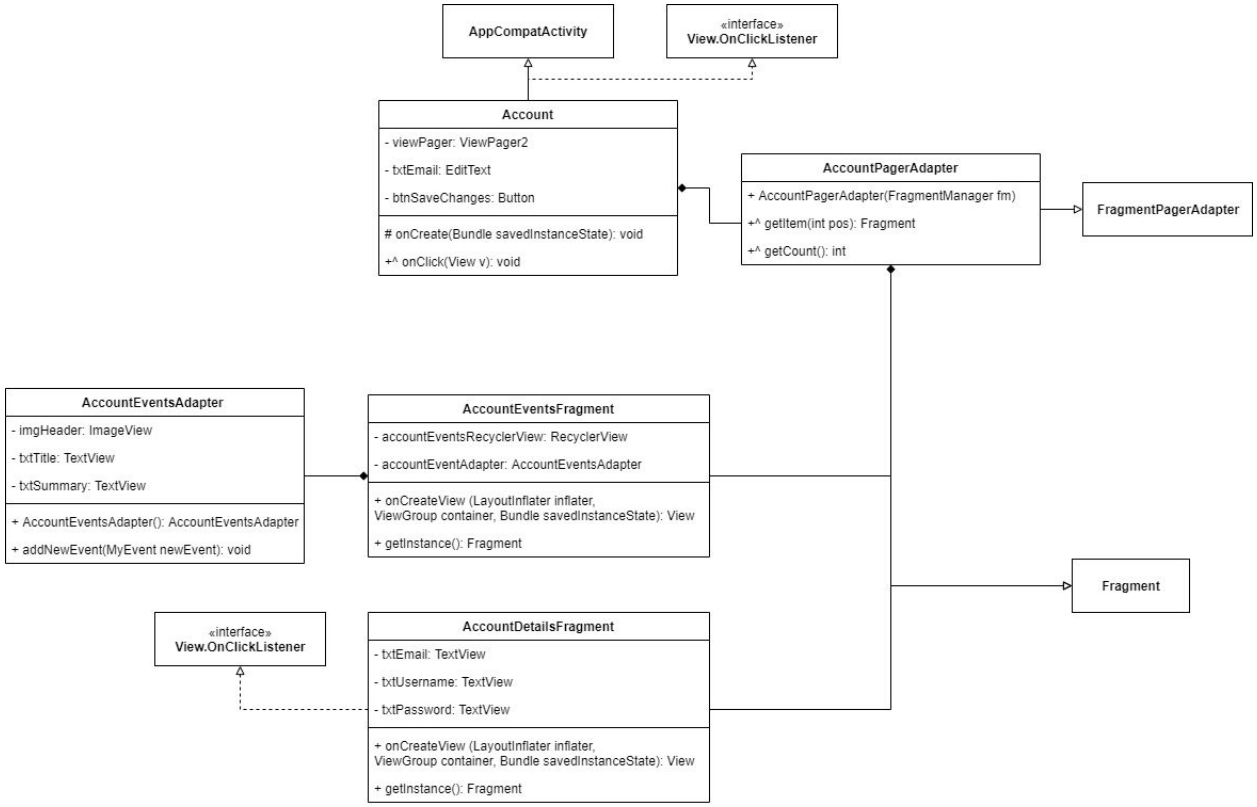
4.1.5 Account View

- **Responsibilities**

  The main responsibility of this view is to display the current login user's information. Including buttons that help users to change their current account.

  Here the use-cases …

- **Class Diagram:**

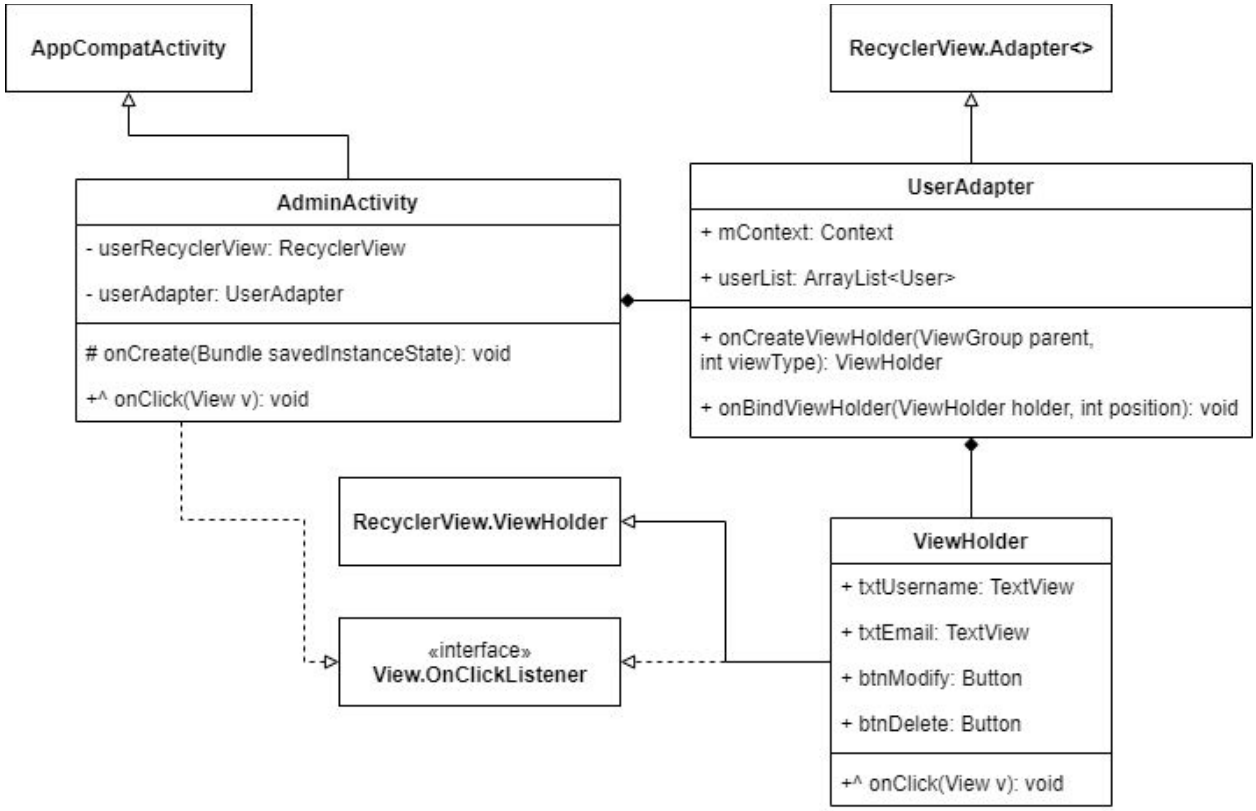


4.1.6 Admin View

- **Responsibilities**

  Display a list of registered accounts with buttons handling add,modify,delete a user.

- **Class Diagram:**



4.2 Component: Controllers

Controllers are the components that handle user interaction, work with the model, and ultimately select a view to render. The view only displays information; the controller handles and responds to user input and interaction. The controller is the initial entry point, and is responsible for selecting which model types to work with and which view to render

4.2.1 Inner Controller

- **Responsibilities**

  Handle all the inner interaction between user and application like (click event, encapsulate the data when user creates an account or event,...)

4.2.2 API Controller

- **Responsibilities**

  Make API calls from the application to the server when the user interacts (load feeds, send post request , get request, …)

4.3 Component: Models

- **Responsibilities**

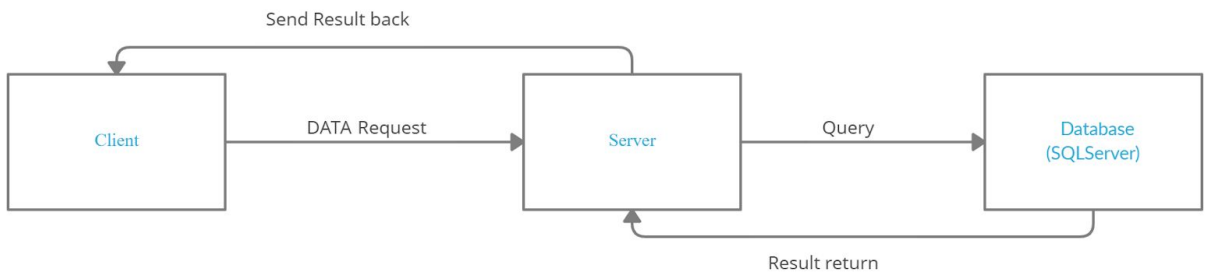
  The Models represent the state of the application and any business logic or operations that should be performed by it. Business logic should be encapsulated in the model, along with any implementation logic for persisting the state of the application. Strongly-typed views typically use ViewModel types designed to contain the data to display on that view. The controller creates and populates these ViewModel instances from the model.

4.4 Component: Database

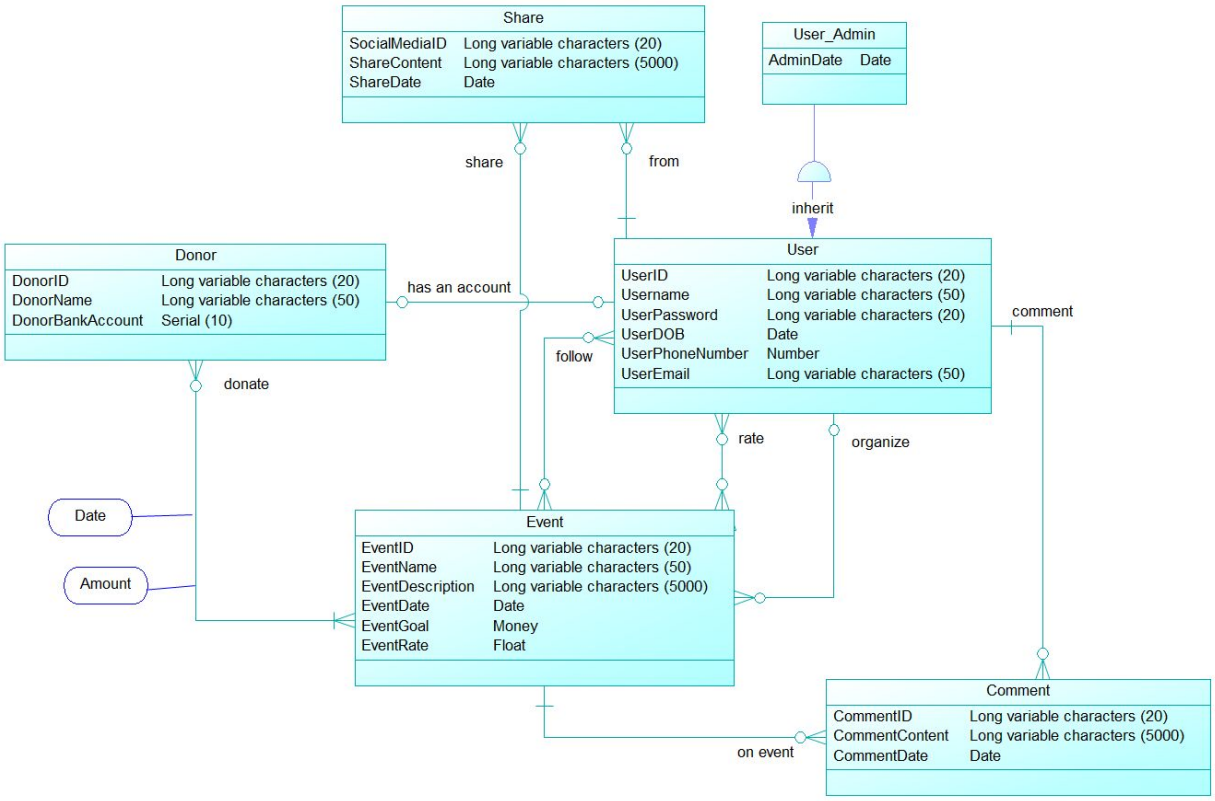
Using SQL Server

- **Responsibilities**

  Store all the data about the application, including donation event, user account, etc



- **ER Diagram**



5. Deployment

[Leave this section blank for PA4 if you are writing this document for PA3.

In this section, describe how the system is deployed by mapping the components in Section 4 to machines running them. For example, your mobile app is running on a mobile device (Android, iOS, etc), your server runs all components on the server side including the database]

6. Implementation View

[Leave this section blank for PA4 if you are writing this document for PA3.

In this section, provide folder structures for your code for all components described in Section 4.]