eFinance Design Details

10/10/2021

Scrum Team

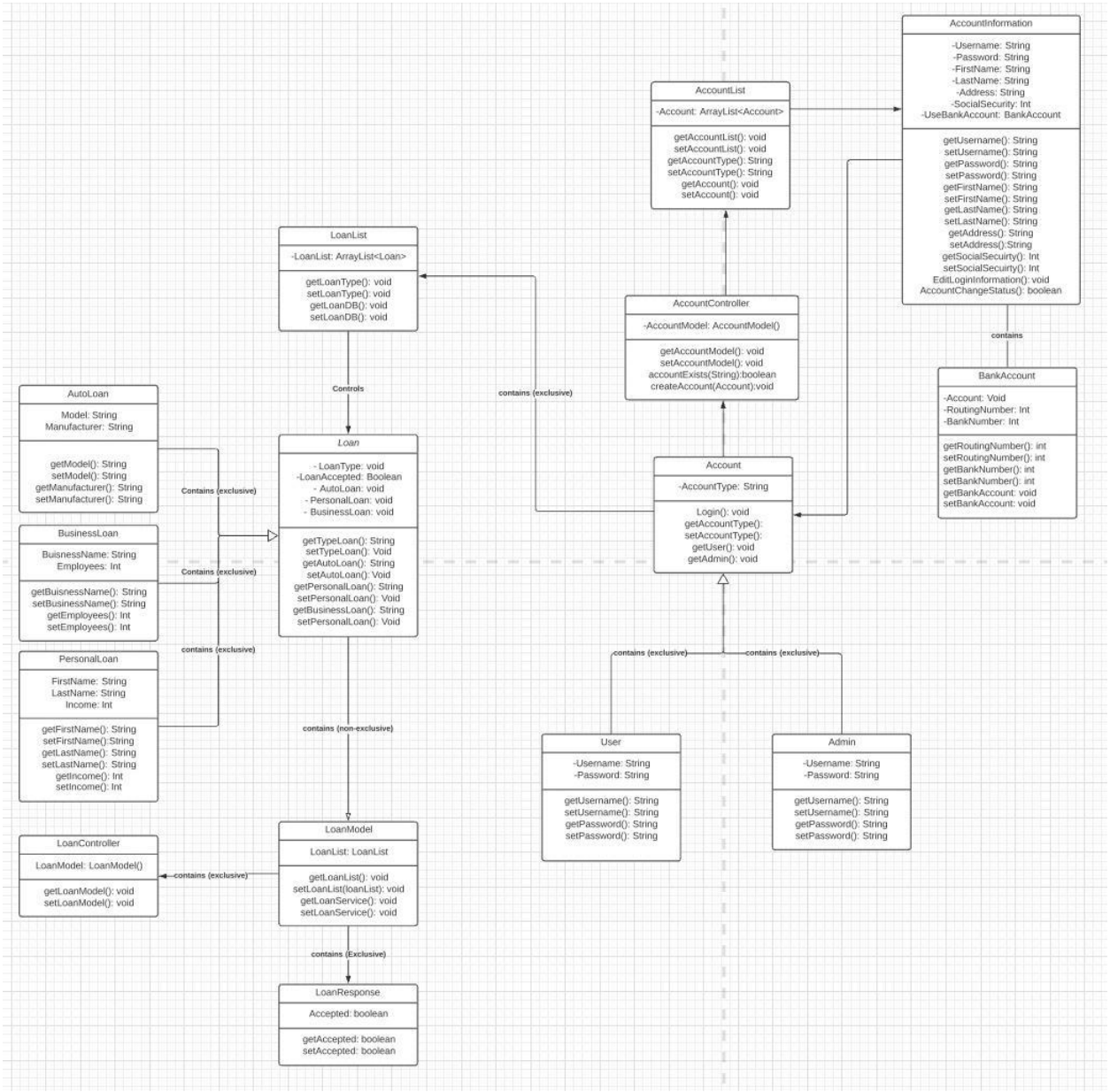
Michael Grillo

Nathaniel Blasco

Jordan Loudis

John Swanson

UML Class Diagram

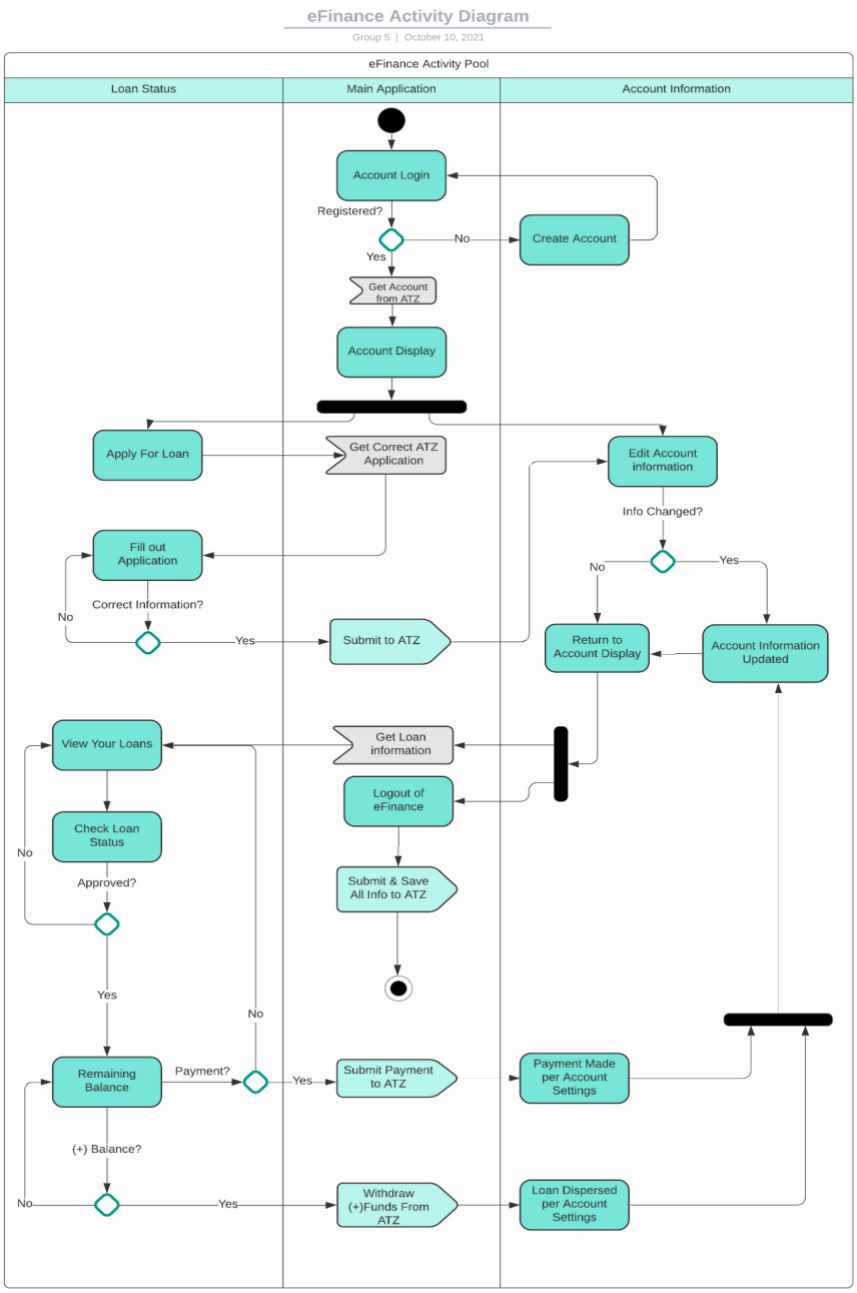


The UML Diagram that we created contains several different classes and methods. Since there is a lot of information, the UML diagram can be more easily viewed if broken up into sections such as our sprints. For example, there is an Edit Account, Loan, Enquiries and Bank Account sprint that are addressed in this UML diagram. First, we will discuss the importance of the create eFinance Account sprint and the classes associated.

To create an account for eFinance there are several different classes and functions that must be addressed in order for the sprint to be successful. First, a person using eFinance must either have a user account or admin account. Account is the one of the main classes in the UML diagram which has a main attribute called AccountType which is implemented to help determine the type of account that is being used. Account contains several other classes such as AccountList, AccountController and AccountInformation. ApplicationController is somewhat self-explanatory as it is the controller for the Account class and controls AccountModel. Lastly, AccountInformation is for personal information of the user or admin of that specific account, such as their name, address, username, and password. Account is also linked to another sprint or topic which is the loan. BankAccount is the last class that is associated with the Account class. BankAccount has several different attributes such as Account, RoutingNumber and BankNumber.

Lastly, LoanList controls the Loan class which has several different attributes detailing the different types of loans available. AutoLoan, BuisnessLoan and PersonalLoan are all classes that extend from the Loan class and allow the user to choose the correct loan type. The Loan class also contains the LoanModel, LoanController, and LoanResponse. LoanModel is the logic for the loan class and has LoanList as an attribute. LoanController is implemented to control the LoanModel and LoanView attributes in the Loan class.

Activity Diagram

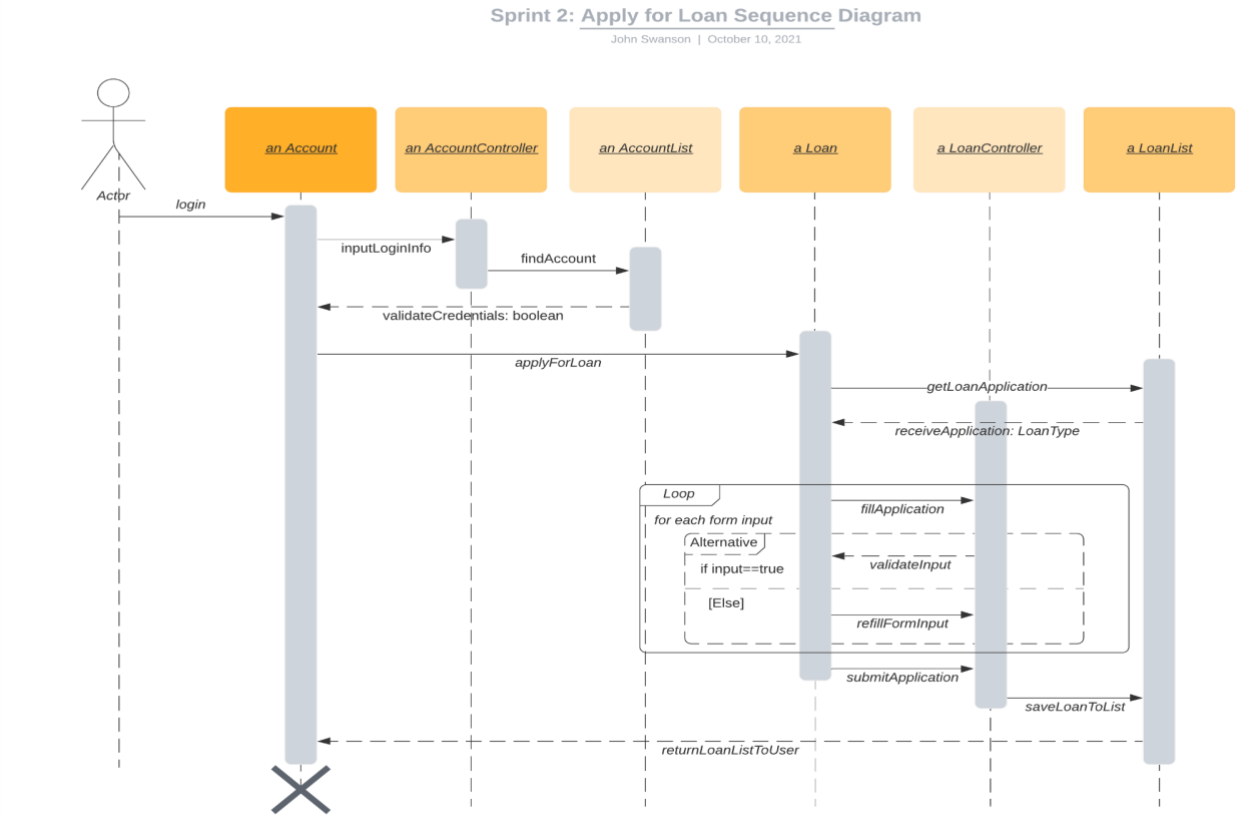


The activity diagram is important to map the flow of the application and the user's experience. This diagram helps the viewer see use of the software through a progression of user

actions. This diagram is split into three sections: Main Application, Loan Status and Account information. The flow of this diagram allows the user to view the behavior of the application.

First, the user will launch the application and will be asked if they have an account. If they do, the Account will then be displayed and several different actions can occur. The actor can then either Apply for a loan or edit their account information. If the user has not applied for a loan they will need to in order to access their account and edit their information. From edit account information, the user can view their loan information as well as check loan status, check remaining balance, make payments and withdraw funds. Lastly, once the information has been changed the application will return to Account Display.

Sprint #2: Applying for Loan Sequence Diagram

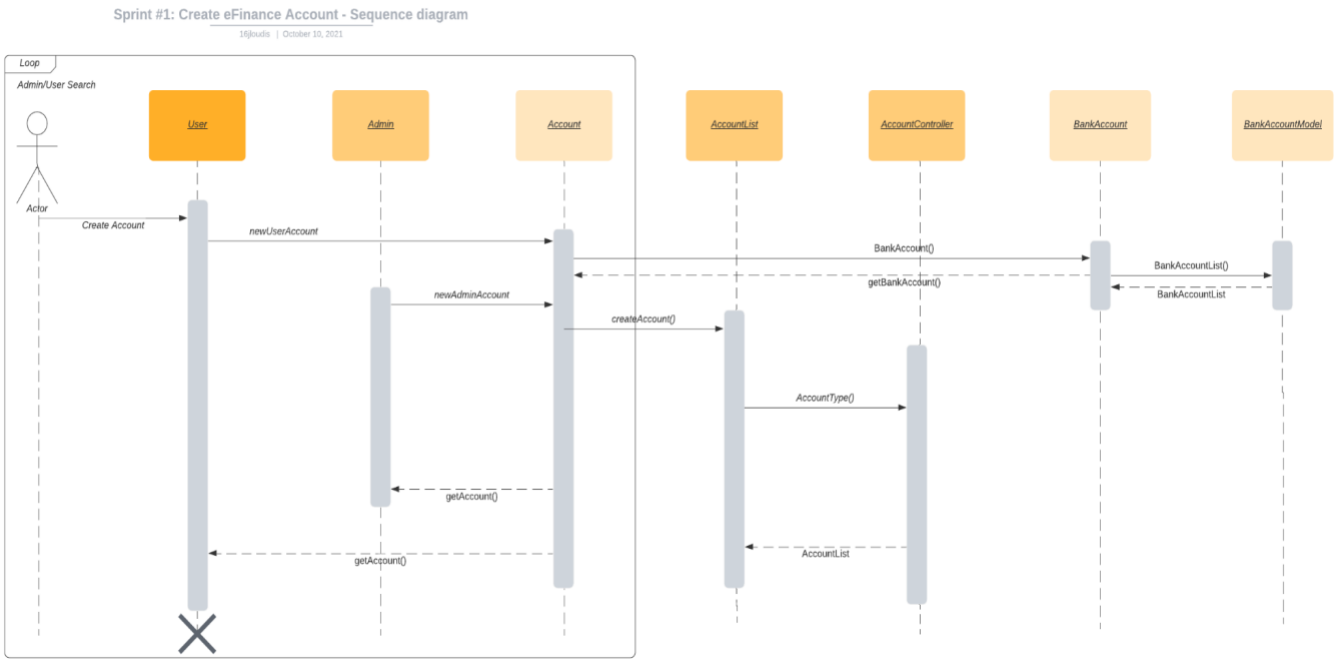


This sequence diagram represents the main functions of Sprint#2. It starts with logging in and validating credentials through the AccountList. After logging in the user is at the main account display view and has the option of clicking to apply for a loan. If the user clicks on the button the loan application is retrieved from the loan list depending on what kind of loan the user is applying for.

The loan application is retrieved, and the user must begin to fill out the application of the loan. If any mistakes are made when inputting data into the loan form and a for loop is created for each one of the inputs in the application where mistakes can occur while applying for the loan. Each input is checked for validation and if not true then the user must reenter the data in that form.

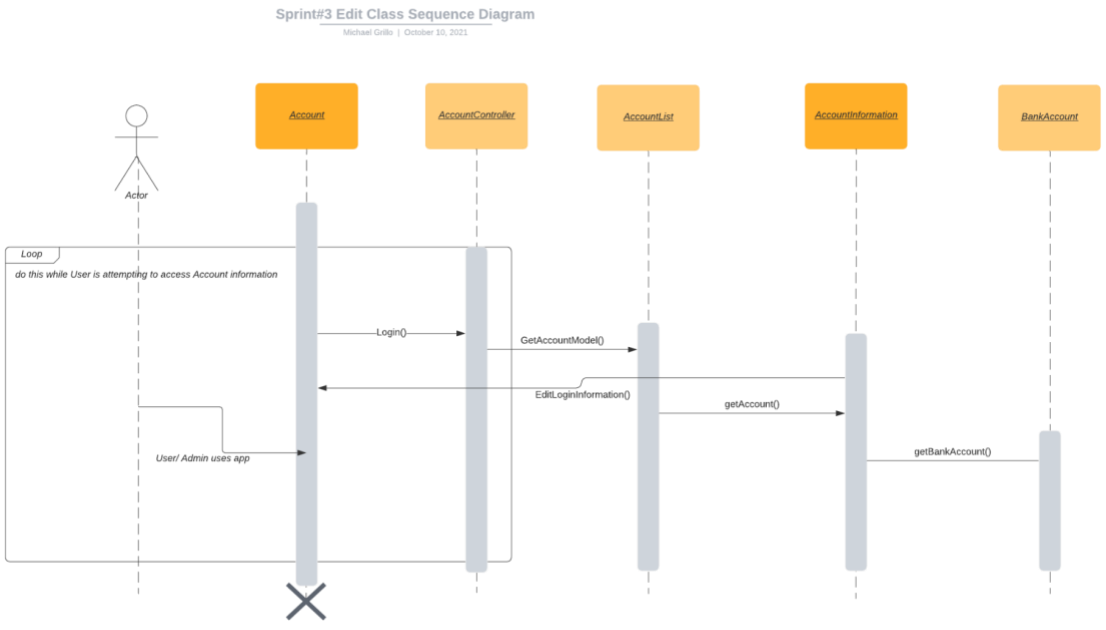
When the loan is submitted, it is saved to the user's loan list class and then sent back to be displayed to the user, which shows each loan's status.

Create eFinance Account - Sequence Diagram:



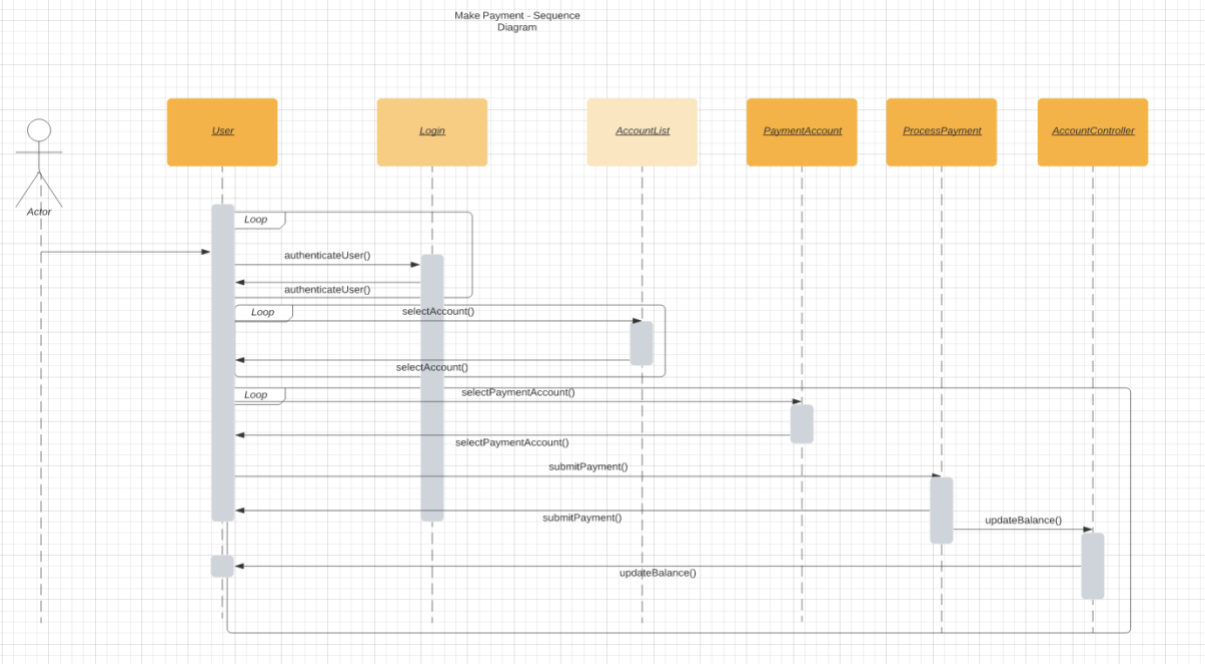
For the create eFinance account sequence diagram there were several classes that must be included for the sprint to function properly. The loop is implemented so the actor is able to search for either a user or admin account and easily create an account. The classes include User, Admin, Account, AccountList, AccountController, BankAccount and BankAccountModel. In order to create a new account, the actor must first either select a User or Admin account to create. After the actor selects their correct account, they will use a username and password to create an account. The Account class then creates a new class to be added to the AccountList class. The Account class is also linked to the BankAccount class which is needed to create an account for eFinance. A method in the AccountList class called AccountType() is then used to determine the difference between the actors and their types of accounts. Lastly, The AccountList can then be controlled by the AccountController.

Edit eFinance Account- Sequence Diagram:



For the edit eFinance account sequence diagram this is a straightforward sprint in which the User will be able to access and edit their eFinance Account information through methods like setFirstName() which are accessed within the AccountInformation class. The User or Admin if the Actor has proper credentials will be able to acquire access to these classes through Identity Access Modifiers determined by their login information provided through the login() method. In our application the content delivered will be controlled through a model view controller architecture which will be expanded upon in future updates. Essentially, the controller will determine what model is available to be accessed based on the login information provided. The ability to edit the account will be delivered within the model if the controller delivers that content. In this sequence diagram this kind of content is defined as EditLoginInformation(); however, there are other methods like setLastName() which allow the user to change the last name listed in the user's Account.

Make Payment for eFinance Account - Sequence Diagram:



For the make payment sequence diagram, the user will be authenticated using authenticateUser() before using a method like selectAccount() to select the loan account from AccountList to make a payment on. Once the user has selected the desired loan account, the user will use a method such as selectPaymentAccount() to select the payment account that will be used to make the payment. After both the loan account and payment account have been selected, a method such as submitPayment() will be used to ProcessPayment. Once the payment has been completed a method such as updateBalance() will be called to update the balance of the loan account.