Role - Developer

The Developers are the programming engines that run the team. Their task is to create high-quality code, based on the requirements/ user-stories and compliant with the Architect's vision which can pass the User Acceptance Criteria set by the team (Bas and testing team). They follow instructions by the TL and also work with the PM to provide reasonable estimates for their interaction with the Customer-facing part of the team.

They will use design patterns and best-practices to create quality functionality that complies with the architectural guidelines set with the nonfunctional aspects of the requirements including, performance, security, speed, etc.

Revision History

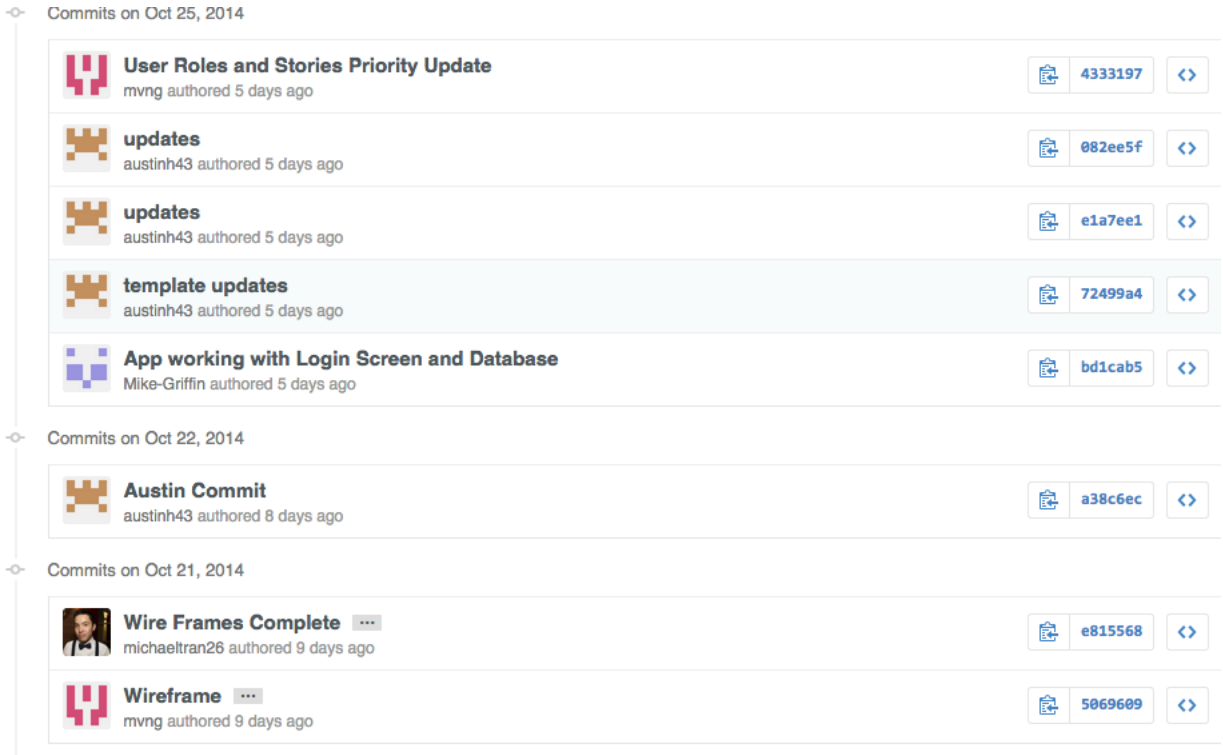
Week Number	Author	Description of changes
1	Jason Dizon Chesong Lee	Finalize database and platform we will be using. Researching android development. Start login page of application.
4	Jason Dizon Chesong Lee	Completed the welcome page/login page. User is now able to create an account and their information is stored onto the Parse database. The user is also able to log back in with their account. We are going to start implementing the debit/credit/show balance of the application.
7	Eden Lin John Chan	Separated user into customer and teller. Improved user interface. Added transfer and close account functionalities. Started working on interest calculation module and penalty module.

Comments: Discuss about the implementation of the debit/credit/show balance structure and how we are going to make sure that people cannot change their balance by malicious injections or through bugs.

- *NOTE: for this document only! The revision history might simply be a git log of your respository*
- *This page should be updated each week and the TAs will use it to determine the week's work done by the team-member in the particular role.*
- *For each of the items mentioned in the following pages. Be as brief as possible in your responses. A good rule of thumb is to keep each response within two paragraphs.*

- This document represents your log of decisions. You are not bound to follow a decision blindly. You may change the decision if new aspects come to light which make your decision inappropriate. However, this may include repercussions like code rewrite etc. So choose wisely.

- You may delete the commented regions for your weekly turn-in submissions.



Commits on Oct 21, 2014

Wire Frames Complete ···
michaeltran26 authored 9 days ago — e815568

Wireframe ···
mvng authored 9 days ago — 5069609

Commits on Oct 20, 2014

Made a successful login ···
KRUKUSA authored 10 days ago — 810bd27

Added Wire Framing
mvng authored 10 days ago — 2002a3e

Honey Badger App ···
KRUKUSA authored 10 days ago — aab9d43

Commits on Oct 18, 2014

Project Manager ···
austinh43 authored 12 days ago — 424c4e2

Finished BA template and added User Stories + Roles doc
jmchan100 authored 12 days ago — 0b49960

Team Lead template for Week 2
Mike-Griffin authored 12 days ago — a4e0840

Added Tester Template
mvng authored 12 days ago — 317105f

Finished Developer Template for Week One ···
jasondizon authored 12 days ago — 237459e

Architect - Week 2 ···
vmannina authored 12 days ago — d0d0e80

Architect Role ···
vmannina authored 12 days ago — 5ebaf92

week 1 developer template ···
jasondizon authored 12 days ago — ef932ed

deleted test file ···
KRUKUSA authored 12 days ago — a76b9ad

Added test file ···
KRUKUSA authored 12 days ago — 46aec82

1. *Coding plan based on the development model chosen by the PM. Also speaking to the architect to design the structure of the credit/debit/show balance page. The tester will begin testing the login page/create account page.*

   Begin coding the credit/debit/show balance page.

   Developers may need to restrict the login/create account page depending on the architect's structure and tester's comments.

   - Don't Repeat Yourself is used in Customer class, Account class, and Teller class. These three classes allow us to query and interact with the database with some helper functions.
   - Single Responsibility Principle is used in decoupling interest calculation from the account class. We have InterestCalculator class specifically for calculating interests and account will be able to use it to calculate the interest.
   - Design Patterns used:
     - Strategy Pattern: we use it to allow different interest calculation algorithm to be used depending on the type of account which is done through the InterestCalculator interface.
     - Model View Controller: Model is the Parse database. View is the activities that user use to perform some operations with their accounts. Controller is the Customer class and Teller class which are used to communicate with Parse database (Model).
     - Factory Pattern: The Parse API we are using utilizes factory pattern which hide the creation process of the Parse object and return back an object that we need without knowing its type.
   - Coding Plan: We follow the UML diagram provided from our Architect to create helper classes and use them to fulfill the design pattern we are using. The user interface improvement is done by discussing with Business Analyst to understand better of what customer wants to see.

2. *Code*

Researching android development.

Login Page

Methods:

- *confirmUserId*
- *confirmPassword*
- *isLoginPressed*
- *isSignUpPressed*
- *isForgotPasswordPressed*

Sign Up Page

Methods:

- *getUserName*
- *getUserDOB*
- *getUserEmail*
- *getUserAccountType*
- *getUserID*
- *getUserPassword*
- *isCreateAccountPressed*

Forgot Password Page

Methods:

- *getUserName*
- *getUserEmail*
- *getUserID*

Account Login

Methods:

- *isCreateNewAccountPressed*
- *isLogoutPressed*
- *isAccountPressed*

Account Information

Methods:

- *isLogPressed*

- isTransferPressed

- isCreditPressed

- isDebitPressed

- isWirePressed

- isClosePressed

Account Log

Methods:

- getHistoryLog

Transfer

Methods:

- getTransferFrom

- getTransferTo

- getTransferAmount

- isConfirmPressed

Confirm

Methods:

- DisplayAccountInfo

- DisplayTransferAmount

- isConfirmPressed

- isCancelledPressed

Credit

Methods:

- isCreditByCheckPressed

Credit Photo

Methods:

- getCheckPhoto

Check Photo Confirm

Methods:

- getCheckAmount

- displayCheckPhotoPreview

- isConfirmedPressed

- is Cancelled Pressed

Wire

Methods:

- getWireFrom

- getWireTo

- getWireAmount

- isConfirmPressed

Debit

Methods:

- getWithdrawalAmount

- isConfirmedPressed

Debit Confirm

Methods:

- displayWithdrawalAmount

- isConfirmedPressed

Close

Methods:

- isBalanceZero

- getUserPassword

- isConfirmedPressed

- Design Pattern UML mapping:
  - Strategy Pattern: Context: Account class; Strategy: InterestCalculator Interface; Concrete Strategy: CheckingInterestCalculator Class, SavingInterestCalculator.
  - MVC: Model: Parse Database such as ParseUser Class, Parse AccountClass, and etc.; View: Various of activities such as AccountInfo activity, Login activity, and etc.; Controller: User Interface which has two concrete classes: Customer Class and Teller Class.
  - Factory Pattern: Creator: The classes that extends ParseObject Class such as Parse User Class, ParseAccount Class, and etc. Product: The ParseObject returned from query call.

Version	Features
1.0	- Login page
1.1	- Login/Create page completed. - Open account. - Credit, Debit. - Show balance page with beautiful user interface and experience.
1.2	- Transfer. - Close account, re-activate account. - Display transaction history. - Divide user into regular customer and teller. - Improve user interface.

CSE110 Iteration Review Questions

NAME: Eden Lin, John Chan
TEAM NAME: Honey Badger
ROLE: Developer

- Is everyone happy with the quality of work? Documentation? Testing?
  Yes. Yes. Yes.

- How did everyone feel about the pace of the iteration? Was it frantic? Reasonable? Boring?
  It was reasonable.

- Is everyone comfortable with the area of the system they were working in?
  Yes.

- Are there any tools particularly helping or hurting productivity? Are there any new tools the team should consider incorporating?
  League of Legends and Clash of Clans are hurting our productivity. Creately helps us improve the productivity. No, currently we are not going to add new tools.

- Was the process effective? Were any reviews conducted? Were they effective? Are there any process changes to consider?
  Yes, it was. Yes, from the check-off. Yes, there were.

- Was there any code identified that should be revisited, refactored or rewritten?
  Yes, we plan to rebuild the project to apply design pattern.

- Were any performance problems identified?
  Yes, the query speed is not as fast as expected.

- Were any bugs identified that must be discussed before prioritization?
  Yes, the application crashes after several transactions.

- Was testing effective? Is our test coverage high enough for everyone to have confidence in the system?
  Yes, it was. We cover most of edge cases in order to make sure that we catch all bugs.

- Is deployment of the system under control? Is it repeatable?
  Yes, it is. Yes, we can repeat the deployment with no issue.