Role - TECHNICAL LEAD

The TL is the leader who organizes the teams technical activities and drives the development of technological artifacts by working with the project manager to divide the tasks between the development team. This role may be one and the same with the Lead Architect on the project.

Revision History

Week Number	Author	Description of changes
2	Mike Griffin	Chose our platform to develop the app, the environment to develop in, and the server and database tools.
7	Chesong Lee	Made sure the development of the product is on task. Made sure that the third iteration of Honey Badger app was on schedule. Made sure the second iteration of the app was tested by the tester and made sure that the fixes were made.

Comments:

- *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.*

-

1. Development Team Technical Proficiency plan

Comments:

- *The TL will be responsible for ensuring that the development team has a common platform to avoid portability issues. They should ensure that the installation of various tools etc are done according to a common specification.*
- *The development team may possess varying level of proficiency with the technology options chosen by the PM and BA teams. The TL is responsible for ensuring that the dev team finds a common ground and is ready to perform when the project enters coding phase.*

2. Onboarding/ Zero-day bootup plan

Comments:

- *This is a simple checklist which allows new members joining the project to come up to speed with the project on their first day itself.*
- *It typically includes:*
  - *link to repositories (git, svn etc.)*
  - *installation instructions for tools or platforms*
  - *Common build and test commands for environment sanity check and simple unit test iteration.*
  - *Project Artifacts that the newly onboarded team member will need: project requirements, architectural decisions etc.*

This document avoids the need for a full-time mentor to be assigned to each new member on the team and it will help when the roles transition between the team members.

3. Technical artifacts delivery plan

Comments:

- *This should be in-tune with the PM's estimation plan and include time for testing.*

4. Version control system and adaptation plan

Comments:

- *How is the team going to manage code and versions? Which versioning tool will you be using?*
- *Is the team comfortable with the chosen system? If git then you can try using pcottle.github.io/learnGitBranching/?demo as a learning resource*
- *What will be the team's production and release cycle?*

1. Our technical plans include creating an Android app and using Eclipse to write for that. We will use Parse for our database and server needs.
2. Zero day bootup plan:

Git repository: https://github.com/Mike-Griffin/Honey-Badger-110-project.git

Installation instructions:

Install eclipse adt with android sdk for your operating system.

Once that is installed go to the virtual device manager. Create a Nexus 4 (arbitrary we may change it later but just to be consistent am saying Nexus 4 for now)

Go to android sdk manager and install all of the packages.

Make sure hello world is running properly.

3. The technical artifacts delivery plan is detailed within the user stories. Certain tasks have a higher priority so they will be completed first. When the database and UI frameworks are complete it will be easier to estimate how the other tasks fit in with that.

4. The version control system we are using is git. We have a repository on github. Production and release cycle has yet to be determined. Will have goals each week to complete.

Chesong Lee

1) The team decided to stick to the platform we are using. We are continuing to use Eclipse to write our Android application and Parse for our database. We begin to add another tool to help us enhance our application, which was Azure, a mobile application UI design program. We had many options when design the UI of the app such as using Adobe Illustrator and storyboards by hand, but our common ground was that we wanted it to be easy to use and fast so Azure was the best tool for that.
2) Git Repository:

   Go to https://github.com/Mike-Griffin/Honey-Badger-110-project.git

   Download in Zip

   Open Eclipse and import the files

   Azure download instructions:

   Go to this website: http://azure.microsoft.com/en-us/downloads/

   Under **Mobile** section click iOS install

   Once the installation is complete, open the program and click new

   Drag and drop the pre-set builders to build the user interface of the application

How to merge Git repo:

https://help.github.com/articles/merging-branches/

Things to test for:

Passed Failed Not Yet Tested

Make sure the login only accepts valid characters – 11/13/14

Make sure password accepts valid characters – 11/13/14

Email is valid – 11/13/14

Make sure the user cannot withdraw over the limit – 11/20/14

Teller's transfer should not exceed the limit – 11/21/14

When user closes account the object is deleted in the database – 11/21/14

Note: Should the account be completed deleted?

Will seek clarification from the customer

Teller can delete accounts and will not be held in the database

Project artifacts:

Requirements: Eclipse with API level 19+

Android Application only

Access to Git repository

Access to Parse database

3)

Create/Open Account	2 Days
Securely Login	1 Day
Check the Balance on all Accounts	3 Days
Print account Statement	3 Days
Iteration 1 - TESTING	**3 Days**

Debit Account	5 Days
Credit Account	5 Days
Transfer Funds	3 Days
Close Account	1 Day
Iteration 2 – TESTING	**5 Days**
Calculate Penalty	4 Days
Calculate Interest	4 Days
Iteration 3 - TESTING	**8 days**

4) Using Github to manage all our code and Parse to manage our database

Team is very comfortable with Github and some people have experience with integrating with databases so it should be no trouble.

Pre-Alpha: 10/18/14

1) Create and Open Account
2) Securely login to your account
3) Debugging/Testing

Alpha: 11/01/14

1) Check the balance on all accounts
2) Debit/Credit Account (Withdrawal/Deposit)
3) Transfer Funds (Between accounts)
4) Close Account

Beta launch will be: 11/22/14

1) Create Teller Account
2) Calculate Interest
3) Calculate Penalty
4) Make UI intuitive

Release Candidate: 12/13/14

CSE110 Iteration Review Questions

NAME: Chesong Lee
TEAM NAME: Honey Badgers
ROLE: Team Lead

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

  Yes everyone is happy with the quality of the work. The documentation was very detailed which made it a lot easier to understand. We wish we had done more testing, but we were satisfied with the given amount of time.

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

  The iteration felt a bit rush, but we were just able to finish it in time.

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

  Yes. If they weren't before, we made sure that the team helped them out to make the person feel more comfortable.

- Are there any tools particularly helping or hurting productivity? Are there any new tools the team should consider incorporating?

  Github is really helping us and azure is really helping us layout the UI of the application.

- Was the process effective? Were any reviews conducted? Were they effective? Are there any process changes to consider?

  Yes the process was effective.

- Was there any code identified that should be revisited, refactored or rewritten?

  Yes, the structure of the program was not OOP in the beginning.

- Were any performance problems identified?

  No

- Were any bugs identified that must be discussed before prioritization?

  No

- Was testing effective? Is our test coverage high enough for everyone to have confidence in the system?

  Yes testing was effective and we did test a lot of edge cases. One way we did test was pre-tend we were the customer and see how we would use the application.

- Is deployment of the system under control? Is it repeatable?

  Yes it is under control and repeatable.