Coupons Nao!

Project Legacy

COP4331_001, Spring 2011

Team Name: A-Team

Team Members:

- Stephen Bryant - sbryant31@gmail.com
- Taylor Kourim - tkourim@knights.ucf.edu
- Daniel Kaplan - kaplan@knights.ucf.edu
- Isaiah Walker - senthose@gmail.com
- Chris Nergard - cnergard@gmail.com

Roles

- Concept of Operations (add in percentages)
  - Taylor Kourim – 50%
  - Stephen Bryant – 50%
- Project Plan
  - Taylor Kourim – 50%
  - Daniel Kaplan – 50%
- Software Requirement Specification
  - Stephen Bryant – 50%
  - Daniel Kaplan – 50%
- Project Management
  - Taylor Kourim - 70%
  - Stephen Bryant – 30%
- Project Design
  - Stephen Bryant – 45%
  - Daniel Kaplan – 45%
  - Taylor Kourim – 10%
- Test Plan
  - Isaiah Walker – 50%
  - Chris Nergard – 50%
- Code
  - Taylor Kourim – 20%
  - Stephen Bryant – 20%
  - Daniel Kaplan – 20%
  - Isaiah Walker – 20%
  - Chris Nergard – 20%
- Configuration Management
  - Stephen Bryant – 20%
  - Daniel Kaplan – 20%
  - Taylor Kourim – 20%
  - Isaiah Walker – 20%
  - Chris Nergard – 20%
- Test Results
  - Isaiah Walker – 50%
  - Chris Nergard – 50%

- User Manual
    - Taylor Kourim – 100%
- Build Instructions
    - Isaiah Walker – 100%
- Project Legacy
    - Taylor Kourim – 20%
    - Stephen Bryant – 20%
    - Daniel Kaplan – 20%
    - Isaiah Walker – 20%
    - Chris Nergard – 20%
-

Analysis

<Analyze all the process and product metrics for your group's project. Combine this information with your understanding of what actually occurred during the project. Based on this information, provide the following analysis (A minimum of one or two well-thought sentences must be provided for each question listed under each bullet. Note that the content of this document must demonstrate a depth of understanding of your project's process and products -- your successes and your challenges.):>

- Assessment of the Quality of the Final Product:
    - The product does work
    - The product works well because there are no loading screens yet does not time out (everything loads in less than 5 seconds)
    - All core functionality has been implemented, although the graphics could be more engaging
    - The application works as long as it is running on an Android device or emulator and has a valid Internet & GPS connection (either via Wi-Fi or 3G)
- Recommended Use of the Final Product:
    - The user manual is perfect to use when you have trouble understanding anything inside of the application, or have troubleshooting problems
    - The application itself is most useful in a scenario where the user is around many shops and plans to do a lot of shopping
    - The build instructions can be used to see how to download our application from the Android Marketplace.
- Known Problems:
    - Application crashes when no location is found
    - Certain characters, including dashes, don't display properly in Coupon Description
    - Change Radius function is not done
    - Sort By will randomly change the order the items are sorted in
    - Google Maps integration was completely scrapped
- Adherence to Project Plan:
    - Our project plan was highly flexible. We frequently modified our plan as different issues arose in order to focus solely on the core functionality.
    - We grossly underestimated how long development would take.
    - The root cause of deviations from our estimates were underestimating the amount of research it would take to learn the Android API and how to get all of the components to interact with each other. Many classes took a long time to write because we did not understand how to express what we wanted to do in a manner native to Android.
    - Miscalculation in the amount of time each part of the project would take led to bottlenecks we could not avoid
    - However, we did finish the project and all of the final documentation on time in the end.
- Defect Analysis:
    - Early Defects:
        - XML wasn't parsing correctly (introduced)
        - Application crashed when trying to transfer activities (discovered & corrected)

        - POST method only works for Strings (discovered & corrected)
    - Middle Stage Defects:
        - Dynamically populating coupon and company list from array list using information from database (discovered & corrected)
        - Uploading images to the SQL (discovered & corrected)
        - Rating problems (introduced)
        - Location threading problems (introduced)
        - Gaining user location without sacrificing performance (introduced)
    - Late Defects:
        - Resizing images (discovered & corrected)
        - Sorting coupons did not always work (discovered & corrected)
        - Populating the autocomplete EditText (introduced)
        - Getting image from gallery returned null pointer (introduced)
        - Getting image from gallery threw IOException (introduced)
- Quality Assurance:
    - We spent a lot of time programming many try/catches to stop the program from crashing in virtually every situation
    - The server checks all input from Add Coupon prior to adding the coupon in the database
    - The Add Coupon activity does some checks to make sure the coupon is valid
    - The application does not continuously poll the user for location in order to increase performance
    - The QA checks all came at the end of development when most everything was completed
    - We could have been more proactive in implementing QA checks and doing QA testing.
- Configuration Management:
    - We used Subversion for our configuration management. It was sufficient, but we frequently encountered many technical issues that were difficult to resolve due to our lack of experience.
    - We could have tried harder to not overlap on the classes we worked on to minimize conflicts
- Suggestions for the Future:
    - Start programming earlier
    - Don't underestimate the amount of effort/time it will take
    - Try to start researching earlier, as far back as the design process
    - Spend more time on the high level and detailed designs, try to think of every possibility ahead of time
    - Have a well documented class diagram and make sure it was followed
    - Processes we would change
        - Spend more time developing the high level design and detailed design
        - More time researching, starting research earlier
        - Meet more often to develop
        - Develop individually more often
        - Assign homework in between meetings
    - Processes we would keep
        - Frequent group programming meetings
        - Paired programming
        - Frequent research
    - Same would go for 100x the scope (each item just becomes more important)