Cook-E development team process

January 19, 2016

1 Software toolset

1.1 Mobile application

The mobile application will be written in Java and will run on Android. It will support Android versions 4.0 and higher (API levels 14 and higher). According to Google's statistics, this will make the application compatible with 96Android devices that have recently provided their information to Google.

The development team chose to target Android because it is widely used and some team members have experience developing Android applications. Java is the primary language used for Android development, and all team members have experience with it.

1.2 Version control

The project will use Git for version control, with central repositories hosted using on github.com.

The project will use the issue tracking and wiki features of the GitHub repository for bug tracking and development documentation.

1.3 Testing

All software components should be accompanied by unit tests developed in parallel. Each team member should write tests for their code. The testing coordinator (see below) will develop integration tests and user interface tests for the application.

2 Group Dynamics

2.1 Core Roles

Some team members have core roles that give them particular responsibilities in the project. All team members will still work on developing the application.

Project Manager: Xianzhe Peng The responsibilities of the project manager include:

- Leading team meetings
- Resolving disputes (see Dispute Resolution below)
- Updating the System Requirements Specification and System Design Specification and Plan for later deliverables

Testing Coordinator: Zijin Zhang The responsibilities of the testing coordinator include:

- Helping other team members set up unit testing for their code
- Writing integration and user interface tests
- Handling regressions, either by making fixes for small problems or opening issues and alerting the responsible team members for larger problems

Infrastructure Coordinator: Sam Crow The responsibilities of the infrastructure coordinator include:

- Maintaining the repository and continuous integration system
- Helping other team members use the version control system
- Checking pull requests
  - Confirming that new code passes tests and follows the code style guidelines
  - Merging code from pull requests
- Maintaining development documentation

User Documentation Coordinator: Carson Lipscomb The responsibilities of the user documentation coordinator include:

- Defining the structure of user documentation
- Ensuring that user documentation is consistent and of sufficient quality
- Before each release, checking that all user documentation is accurate

2.2 Dispute Resolution

Features After the core features in the software requirements are finalized, no feature may be added to the project unless five of the seven team members vote in favor of adding it.

At any time, the project manager may decide to remove a feature to ensure that the project is completed on time.

Other Disputes If a dispute cannot be resolved through deliberation, the project manager should make an executive decision to resolve the dispute. The decision may be overridden if five or more of the seven team members vote to override it.

2.3 Justification

This system of roles ensures that important tasks are performed but does not restrict the development tasks that team members can work on. The feature dispute resolution process is designed to reduce the risk of feature creep. The resolution process for other disputes is designed to be fast while still allowing the team to override project manager decisions in unusual circumstances.

3 Timeline

- 2016-01-22: Software requirements specification due
- 9 days: Team splits into user interface design, algorithms, and application groups
  - 3 days: user interface design group develops detailed user interface design, other groups design data and algorithms
  - 6 days: Application group starts implementing user interface, other groups finish data and algorithm design
- 2016-02-01: Software design specification due
- 6 days: Prepare zero-feature release
  - 2016-02-02: Infrastructure coordinator creates project
  - Application group implements user interface
  - Other team members, led by the infrastructure coordinator and the user documentation coordinator, write initial documentation
- 2016-02-08: Zero-feature release due, algorithm design must be complete
- 11 days: Implement features
- 2016-02-19: Beta release due
- 6 days: Prepare for feature-complete release
  - 2016-02-20: Testing coordinator identifies deficiencies in tests and asks the responsible team members to improve them, user documentation coordinator identifies user documentation deficiencies and arranges for them to be fixed
  - 2016-02-25: Testing coordinator runs coverage tool and records results
- 2016-02-26: Feature-complete release due

4 Risk Summary

4.1 Scheduling Algorithm

The scheduling algorithm is the most critical part of this project. It needs to solve a problem with many constraints. The most difficult part of the algorithm is being able to adjust to users' varying times at completing different steps. If users do not perceive that the algorithm is better at scheduling than a person, they will not use the application.

To improve the probability of the team developing a good algorithm, we have allocated a long period of time early in the project for algorithm design. Part of the team will work primarily on the algorithm.

4.2 Acquiring Recipe Data

A key part of this project is acquiring enough recipes to be useful for a user who may want to cook any combination of dishes. There may be difficulty in finding data sources for the recipes and then creating an automated way to pull that data and putting it into our own database (or streaming it from the source).

For the early prototypes we will manually add around 10 recipes and then spend time adding recipes to the database after the scheduling algorithm is finished

4.3 Testing Difficulties

Testing the whole application and its user interface will be more complex than small-scale unit testing. The testing coordinator will be responsible for setting up and maintaining user interface tests. If automated user interface testing is not feasible, the testing coordinator will develop procedures and test the user interface manually.