Development Plan

Gimball3000 ; Trillium Bluetooth Event Tracker

Table of Contents:

- 1. Developer Workflow
- 2. Version Control
    - 2.1. Feature Branches
    - 2.2. Code Review and Merging
- 3. Documentation
- 4. Testing
- 5. Continuous Integration

1. Developer Workflow

During development, there will be development tasks common to all developers and tasks tailored to assigned team member roles.

The high-level overview for typical developer workflow will follow this series of events, which will be elaborated on in later sections of this document:

1. Team meets at start of iteration to determine the iteration's feature/requirements scope and assign feature teams.
2. Feature teams create feature branches in version control.
3. Feature teams will assign a branch manager, who will be responsible for periodically merging Master into that branch and any necessary merges from the branch to Master. Any merges from the branch into Master will require a pull request that will need to be reviewed by a team member that has not contributed towards the feature if at all possible.
4. Feature team members checkout that branch and work on items for that feature on that branch, committing as then go. During this time the branch manager will be doing periodic merges as well.
5. On feature completion a pull request will be made to request the branch be merged into master.
6. Team members that did not work on the feature will review the pull request, comment on it and +1 it if it is ready to merge in.
7. Feature developers will test the functionality on Master to ensure it still works.
8. Once a feature branch has been merged in successfully and all work is done on the feature, the branch will be deleted.

Role-level workflow will be dependent on the role as defined in the scope of this project. Role assignments are defined below:

- Test Lead : Randy

- Configuration Lead : Anshul
- Documentation Lead : Tyler

2. Version Control

The team will be using git for version control and so will follow conventions related to git.

2.1. Feature Branches

Feature teams will manage the creation, development on, merging, and deletion of feature branches. Doing feature development will not be allowed on Master branch, in order to reduce likelihood of conflicts and downtime due to breaking changes.

2.2. Code Review and Merging

The team will be following a lightweight code review process centered around reviewing commits that are logic-heavy or any doing major development work, as well as all pull requests, which will be made when developers want to merge in a feature. Code will not be committed or merged if it has outstanding bugs, breaking changes, or unresolved comments/discussions. Code authors should address all comments with the team member who made the comments and get their approval (+1) before merging in code.

3. Documentation

Documentation will be will be managed through Jazzy a command line utility that generates documentation pages for Swift and Objective-C projects. Documentation should follow XCode quick help guidelines using markdown, which can be found here. This will allow developers to easily read documentation for the project while developing as well as thorough documentation for others who do not have XCode or the codebase available to them at the time. Having well-defined documentation will improve quality of the product and provide an easy way for developers to efficiently learn how different facets of the application communicate and interact with each other. This will also aid future developers that work on the application post hand-off.

https://github.com/realm/jazzy

4. Testing

Tests will be written with each feature and are required to be passing on feature commits/pull requests in order for the commit/pull request to go through. Continuous integration will run the tests in order to ensure quality and report back to reviewers whether tests passed or not. Having passing tests and a stable build is important to ensure quality so developers and reviewers will pay attention to build status and fix any broken tests related to work the developer has been doing or done in the past. If the code author is unable to fix their tests, a delegate developer may attempt to fix the tests for them in order to reduce build downtime.

Testing will include a variety of tests including unit testing, integration testing, and front-end testing. Tests will be meaningful and cover all major code flows and common or breaking edge cases.

5. Continuous Integration

Jenkins will be used as a continuous integration system for both the iOS app and the backend component. It integrates with Github as well and can be configured to run on commits and pull requests to any branch on the repository. It will provide regression test coverage by running any written unit tests, and send results out to developers on Slack/ email. This will make sure that all new pieces of code will pass the test cases before being pushed into production environment.