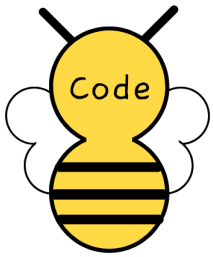
Development Process
CodeBee

Hongqing Cao, 400053625
Jason Nagy, 400055130
Shivaansh Prasann, 400100351
Sida Wang, 400072157
Weidong Yang, 400065354
Yachen Wu, 400131465

November 9, 2020



Contents

List of Tables

List of Figures

Date	Version	Notes
Nov 04, 2020	0.1	Doc created with structure
Nov 07, 2020	1.0	Initial Draft
Nov 08, 2020	1.1	Final Draft

Table 1: **Revision History**

1 Roles and Responsibilities

1.1 Roles

- **Project Manager:** Shivaansh Prasann.
- **Software Developer(s):** Everyone in the team.
- **Software Tester(s):** Everyone will be testing the software components that he/she developed. The integration testing for specific software components will include team members who are responsible for those components.
- **Git Release Coordinator(s):** Everyone in the team will take turns acting as the release coordinator.
- **Meeting Note Taker(s):** Everyone in the team will take turns acting as the note taker for each team meeting.

1.2 Individual Responsibilities

Name	**Responsibilities**
Shivaansh Prasann	• Project Management • Game Level Design & Programming
Sida Wang	• Full-Stack Development • Database management
Yachen Wu	• Front-End Development • UI Design
Hongqing Cao	• Front-End Development • UI Design
Jason Nagy	• Back-End Development • Custom Compiler Design & Implementation
Weidong Yang	• Back-End Development • Game Level Design & Programming

Table 2: **Roles and Responsibilities**

2 Version Control

2.1 Git Branching Model

In order to facilitate development of multiple modules simultaneously, the entire team is expected to follow an AGILE methodology, which will be explained in Section 3.1. Weekly meetings will be held to plan scrums and assign priorities to different tasks. Periodic iterative releases will be aided by the use of GitHub version control to ensure that development, testing and implementation of one feature does not hold up development of other features.

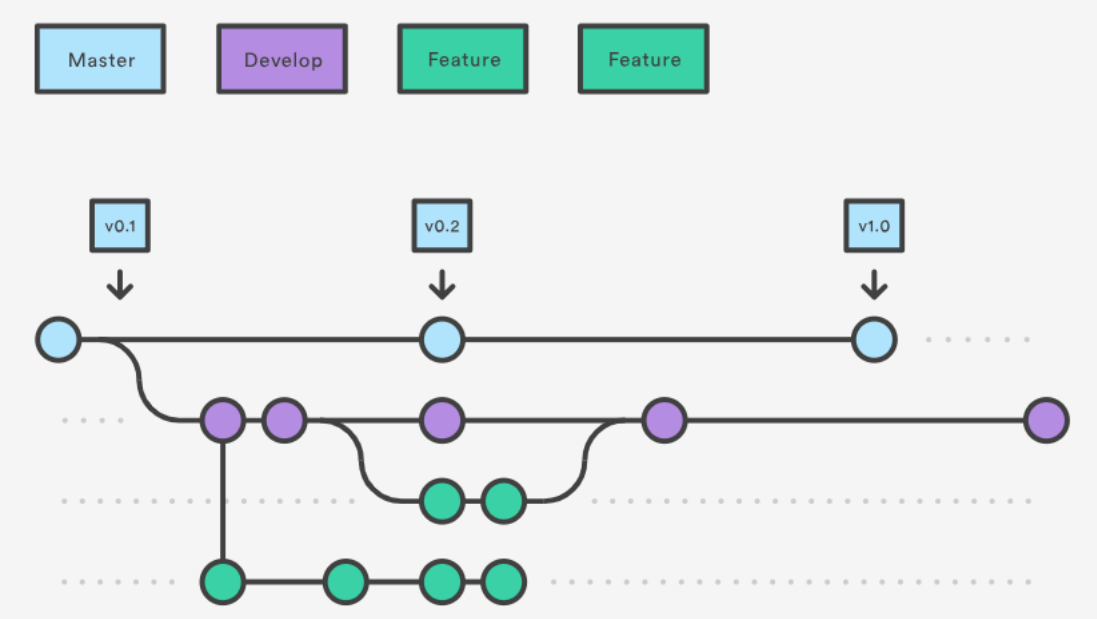


Figure 1: **Git Flow Diagram**

Each team member will make their changes in a "feature" branch which is sourced from the "develop" branch. Once development is complete, unit testing will be performed on the feature branch.
Once unit testing is successfully completed, the changes from feature branches will be merged into the develop branch. Integration testing will then be performed on the develop branch, and if no errors are found, then the master branch will be updated with the changes in the develop branch.

2.2 Process of creating changes

Each team member is required to create an individual "feature" branch which is sourced from the "develop" branch when developing different components of CodeBee. After precisely reviewing other member's feature branches and test results, the feature branch(es) will be merged into the develop branch. Integration testing will be performed on the up-to-date contents of the develop branch using a local Linux server, and once integration testing is considered successful, the master branch will be updated with the contents of the develop

branch. If any bugs are found, they shall be documented in the team's Trello board and will be discussed either immediately or in the next upcoming weekly meeting, depending on the severity of the bug. The team will be encouraged to commit often, and add helpful commit messages to their branches.

3 Process Workflow

In general, for all development and documentation tasks, the following steps shall be followed:

1. The upcoming tasks shall be discussed during a weekly team meeting. Meeting minutes shall be recorded for those who are unable to attend the meeting.
2. The various tasks shall be delegated to different team members according to preference, expertise and availability.
3. To keep track of various action items and assignees, a Trello board will be used. This board shall contain cards for each task, and assignees will be added to these cards. Each card will have notes and a deadline attached for better visibility.
4. For development tasks, each member will create a git branch to work on their tasks, and once their work has been completed and tested, it will be merged to the main branch in the repository. See the 'Version Control' section for more.
5. Changes will be released with the assistance of the release coordinator. Integration testing will be performed on a local Linux server once the changes have been merged to the develop branch.
6. For documentation tasks, the team members shall work on their assigned sections on their own time, and a review meeting shall be held before the document submission deadline.

3.1 Steps to be taken

1. Create functional and non-functional requirements for CodeBee.
2. Create high level design diagrams for CodeBee.
3. Create detailed designs for each module of CodeBee.
4. Implement, test and integrate all models of CodeBee by following an AGILE methodology. This means that we shall have incremental releases for features and bug fixes and will be testing and updating our application throughout the development life cycle (as shown in the diagram). These include the following:

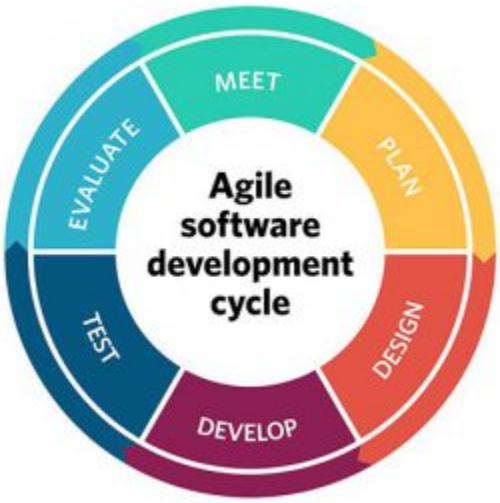


Figure 2: **AGILE Methodology**

(a) Modules containing the front end UI for the application.
(b) Modules containing the functionality for the UI.
(c) Modules containing the logic for playing the game.
(d) Modules containing the logic for game progression.
(e) Modules containing the logic for the achievement system.
(f) Modules containing the user login system functionality.
(g) Modules containing the database functionality.

5. Perform deployment for CodeBee on a Cloud server.
6. Perform usability testing for CodeBee with users.

4 Development Tools

- **Microsoft Teams**: Team communication and meetings.
- **Visual Studio**: Code editing.
- **Atom**: Code editing.
- **Local Server**: Application testing.
- **AWS**: Cloud deployment.
- **JSDoc**: Documentation of Javascript code.
- **LaTex**: Documentation of written content.

- **Trello**: Kanban Model, Tracking of tasks.
- **Web Browser**: Application testing.
- **GitHub**: Version Control.

5 Coding standards

5.1 Naming conventions

- Variables, Functions, and Classes should be named using Camel Case.
- Constants should be named using Capital Letters only.
- The naming of each item should introduce the meaning of it.

5.2 Indentation

- Four spaces should be used to represent one indentation.
- Each nested block should be properly indented and spaced.
- All braces should follow the function/class.

5.3 Comments

- All state and global variables should have their meanings and usage explicitly stated.
- Comments should be written in accordance with JSDoc format for communication of program flow.

6 Handling Changes

6.1 Bug tracking and change request tools used

When a bug is found, it should be be immediately noted into the Trello board in the bug category. Then the bug can be tracked within Trello. During the weekly meetings, a portion of time will be dedicated to discussing bugs and assigning priorities based on the impact of the bug. Only high-impact bugs will be discussed in-depth. Should a team member discover a bug that they think requires immediate attention, they can message the team about this bug using Microsoft Teams.

6.2 Classification of changes

1. Feature: Any new functionalities available to the user will be considered as a 'feature' type change.

2. Bugfix: Any changes which resolve known issues or bugs with the application will be considered a 'bugfix' type change.

3. Maintenance: Any changes, which do not affect the application directly but are required for proper functioning, such as hosting scripts or database connectivity modules, will be considered 'maintenance' type changes.