Elysium: A New Generation Music Sharing Application

`Code Review and Merge Policy`

Garrett Phillips — Benjamin Boardley — Jack Gardner
Purdue University, Elmore Family School of Electrical and Computer Engineering
{gwphilli@purdue.edu — bboardle@purdue.edu — gardne97@purdue.edu}

January 2024

1 Introduction

This document outlines the repository architecture, code review policy, and merge polices intended for the developers of the Elysium project.

2 Repository Architecture

We will be using Git as our Source Control System, managing our directories on Github. Below details the directory tree we are planning on implementing for the duration of our project, Elysium.

```
$ tree -S
.
 Elysium
    pyvenv.cfg
 elysium_backend
    elysium_backend
        __init__.py
        asgi.py
        settings.py
        urls.py
        wsgi.py
    api
    db.sqlite3
    manage.py
 elysium_frontend
    __tests__
    android
    ios
    node_modules
    src
        components
        screens
        navigation
        App.js
        index.js
    assets
    app.json
    package-lock.json
    package.json
```

3 Review and Merge Policy

1. Pre-Commit hooks for linting code

   (a) Flake8 for python (Django)

   (b) ESLint for Javascript (React Native)

2. Merge Policy

   (a) Our merge policy dictates that before a merge is finalized, pull requests, along with passing all unit tests created, must be reviewed and accepted by at least one reviewer other than the author who made the pull request.

3. Branching Policies

   (a) Our repository branch policies will consist of one production branch along with multiple working branches.

   (b) At the lowest level, each issue of our project will have its own feature branch where multiple authors can commit features and changes to the specific issue. These feature branches will be branched off of the development branch.

   (c) Below the production branch will sit a development branch acting as a working directory where all merges and commits from issue branches will gather. It will also serve as an integration point where merge conflicts and the most recent code is located.

   (d) The production branch will only be used for production ready code. Merge requests from the development branch must be accepted by both other persons before the merge is finalized into the Master.