Table 1: Revision History

Date	**Developer(s)**	**Change**
Date1	Name(s)	Description of changes
Date2	Name(s)	Description of changes
...	...	...

SE 3XA3: Development Plan
ohm

Team 4, ohm
Jonathan Brown, brownjs2
Graeme Crawley, crawleg
Ryan Marks, marksr2

Put your introductory blurb here.

1 Team Meeting Plan

2 Team Communication Plan

Our communication plan is to use a Group Facebook chat for all synchronous communication between team members. This was chosen because of the ubiquity of Facebook Messenger and the fact that we all check it more often than we check email. For asynchronous communication GitLab issues will be used for work that is yet to be done, and Gitlab merge requests to review work that has been done. Any communication between team members and people external to the team (Course staff, technical mentors, etc.) should be reported into the Facebook chat, with any concrete action items added as Gitlab issues.

3 Team Member Roles

4 Git Workflow Plan

We will follow a feature branching model for source code and a linear commit process for document changes. In addition we will mandate code review and merge requests to force understanding of all code. Merge requests should be "Thumbs Upped" on Gitlab by both other developers before being merged by the author however this is not a hard rule: If a developer feels they must commit code to master or merge without approval they can, so long as they can feel the circumstances are justifiable. This was chosen over the gitflow model as gitflow's benefits are for tracking a software system with regular releases that must be maintained. As our software is in such early stages that we will not have legacy releases to maintain, the benefits of gitflow are negligible when compared to the

organizational costs. The documents however will use a linear commiting style to master as documents are easier to merge and there is a good deal of value in everyone being aware of unfinished sections of the document.

5 Proof of Concept Demonstration Plan

As a proof of concept, we plan on producing a form of the application with more limited functionality as it will require the appropriate placement of the resistor within the image. The proof of concept will most likely lack the ability to identify resistors in an arbitrary part of the image, a feature which is inteded to be included in the final release.

6 Technology

Ohm will be written in Java, with the desktop and potentially Android as our target platform. Gradle will be used a build tool and dependency manager. OpenCV, an open source image processing and computer vision library, will allow us to perform the required analysis of the input images of resistors.

7 Coding Style

All group members will adhere to the Google style guide for Java outlined here: `https://google.github.io/styleguide/javaguide.html`.

8 Project Schedule

A Gantt Chart that outlines the current project schedule can be found in the ProjectSchedule directory of this repository.

9 Project Review