PX913 report on what each person did individually and what was worked on together:
Link to the GitHub (set up by Fraser): https://github.com/Fraser-Birks/px913-project.git

Fraser:
I completed the following things:
- Writing a function which takes an electrostatic potential and finds the Ex and Ey components of the electric field.
- Creating the particles data type which holds information about each particle, solving for particle accelerations using the electric field, and implementing the velocity-verlet algorithm to solve for particle trajectories and velocities.
-NetCDF file writing for the output data, with the particle information being written along an infinite time dimension each timestep.
-As an extension, I also extended the particle velocity solver functions to be able to work on an array of particle types (such that a solution can be found for multiple particles). I also added an interface to the NetCDF file-writing in order to be able to write an array of particles and a single particle with the same function call.
(NOTE – You can see the multiple particle solver in action if you run the multiple_particle_buildscript in the extra folder provided!)

Chantal:
I completed the following things:
- Writing a module with subroutines to create the three initial charge density states ‘null’, ‘single’ and ‘double’,
- Writing the Gauss-Seidel solver which iteratively solves for the potential at each grid point based on the initial charge density until the normalised error is below a set tolerance
- Implementing the commands to read in the input from the command line and check that each variable has been successfully red in
- Doing the visualisation of the particle path, charge density grid and Ex component of the electric field grid, using matplotlib (besides the animation, which was done collaboratively)

Work completed together:
Initially, we collaborated together to decide the data-structure we planned on using for the project; we decided on using one overall module to hold all of the data structures we need, and for all of the code we individually wrote to USE this module.

Collaboratively, we did some debugging work on the Gauss-Seidel solver, as well as some general code testing and introducing some fail-safes and checks during runtime. We also worked together to integrate our created modules, produce the build script, and on producing the final animation.