**Health tech**

**Lesson 3: Prototyping innovations**

**Introduction**

In this lesson, students work in teams to complete the prototype of their health tech innovation, using a number of computational thinking concepts and skills. The lesson can easily be adjusted in terms of length and teacher input to suit your students experience.

**Time:** @60 minutes

**Materials needed:** Lesson plan, lesson guide, materials for prototype creation (e.g. large and A4 sheets of paper, pens, micro:bits if using and/or access to the MakeCode editor, step tracker example hex file (if using).

**Learning objectives**

* To work effectively as a team to develop a prototype for a health tech innovation
* To design an accurate, detailed algorithm for at least one prototype feature
* To use the algorithm to write, test and debug a working micro:bit program

**Lesson summary**

1. Recap of challenge & team planning (5 minutes)
2. Team work & workshops (40 minutes)
3. Sharing progress (10 minutes)
4. Review & wrap up (5 minutes)

**Introduction: Recap of challenge & team planning (5 minutes)**

* Ensure students have their worksheets and plans from last lesson.
* Spend a few minutes recapping the challenge and criteria for their health tech innovation (**slides 2 and 3**) and asking teams to consider how they will work effectively to complete the challenge.
* If you wish, ask teams to set ‘responsibilities’ and note down objectives for this lesson E.g. overall team leader, visual designers, programmers).
* Ensure they understand they will be presenting their prototype to an audience next lesson and share the learning objectives on **slide 4** if you wish.

**Team work & workshops (40 minutes)**

* Ensure students have access to all that they need to complete their prototype and give them ample time to complete it, offering support and interim ‘time and progress checks’ as appropriate.
* You may wish to run a series of ‘workshops’ with representative from each group responsible for that task. E.g.
  + Discussing plans with the ‘visual designers’ and explaining what a ‘good’ prototype looks like (**slide 5**).
  + Running an ‘algorithm design / programming clinic’ to help students work through any issues they are having as they write their algorithms and programs (use the example on **slides 6 & 7** and the **step tracker example hex code** if helpful).
  + Using **slides 8 and 9** to explain paired programming and reminding students of the importance of testing and debugging.

**Sharing progress (10 minutes)**

* Explain to teams that next lesson they will be presenting their prototype (**slide 10** - adjust as necessary to suit).
* Ask teams to discuss their progress in today’s lesson within the team, giving each member the chance to feedback and discuss what they need to in order to complete their prototype, ensuring they take note of actions for next lesson.
* Ensure teams are sure how much time they to get ready for their presentation in the next lesson and if possible/necessary, offer support prior to the next lesson so this is achievable.

**Review & wrap up (5 minutes)**

* Use **slide 11** to ask students to consider in their teams what computational thinking concepts and skills they have used in today’s lesson and share as a group (e.g. abstraction by creating a representation of their innovation, decomposition by breaking down the challenge into smaller tasks).
* Highlight important skills such as problem solving, team working, sticking to deadlines etc, sharing examples from different teams.
* Review the learning objectives on **slide 12** if you wish.

**Extension ideas:**

* Teams could create a short informal video to explain their prototype development so far and how they have approached the challenge with each member explaining their role, any problems they have encountered and how they have overcome them. and how they have.
* Students from each team with the same ‘role’ could get together and share their progress and discuss challenges.

**Differentiation**

**Support:**

* Students can be given an appropriate role playing to their strengths, or paired with another student if helpful.
* Additional adult support can be given where needed to help them fulfil their role successfully.

**Stretch & challenge:**

* Students can be given more challenging roles according to their strengths and/or areas of development and can write (or support other members of their team to write) more complex algorithms and code.
* Students can set themselves, or be helped to set, stretch and challenge objectives for this and next lesson.

**Opportunities for assessment:**

* Informal observation of individuals and teams during teamwork and workshops.