Child’s Toy Project Plan: Race Car

Introduction

The proposed toy concept for this project is a remote-controlled car. The remote controller will be a software-based application designed with both children and guardians in mind.

The toy will be gender-neutral, considering studies demonstrating that there is overlap in pre-schoolers’ interest in toys regardless of their gender (Fine & Rush, 2018). The design will support these goals by enabling the car’s customisation and configuration of the driver’s appearance. This project will focus on developing the software controlling the car remotely and not on its hardware.

Scope

This project will produce a software application to control the car and demonstrate action in a virtual environment.

Stakeholders

For this project, the stakeholders will be:

- The child playing with the car.
- The child’s guardians.
- The toy producer.

Future stakeholders may include regulatory bodies, safety groups, and academic organisations.

Requirements and Objectives

The following is the breakdown of the requirements split per each user stakeholder:

- **Table 1** for the 'child';
- **Table 2** for the 'guardian;
- **Table 3** for the 'producer'.

***Table 1**. Requirements for the 'child'.*

Child			
#	Requirement	Type	Priority
1	Control race car's movement	Functional	High
2	Control race car's speed	Functional	High
3	Honk the horn	Functional	Low
4	Change race car's LED colours	Non-functional	Low
5	Customise driver's appearance	Non-functional	Medium
6	Display battery status	Non-functional	High
7	Send alert when battery is lower than 20%	Non-functional	Medium

Table 2. *Requirements for the 'guardian'.*

Guardian			
#	Requirement	Type	Priority
1	Password-protected parental control access	Functional	High
2	Limit maximum speed in the parental control menu	Functional	Medium
3	Set driver's appearance	Non-functional	Medium

Table 3. *Requirements for the 'producer'.*

Producer			
#	Requirement	Type	Priority
1	Receive usage-related statistics	Non-functional	High
2	Receive diagnostic reports	Non-functional	Low

These requirements were defined to be SMART (Specific, Measurable, Attainable, Realistic, and Timely) (Bjerke & Renger, 2017; Ogbeiwi, 2017), thus providing guidance to delineate a healthy and valuable product backlog (Gaikwad et al., 2017; Al-Saqqa et al., 2020).

Use cases

Several requirements are interdependent, as shown in **Fig. 1**. Thus, planning must consider this aspect and minimise back-to-back scheduling-related risks:

1. Building the parental control function for speed limitation requires development of basic car controls and the parental control menu.
2. Analytic functions depend on the development of the relative control functionalities.

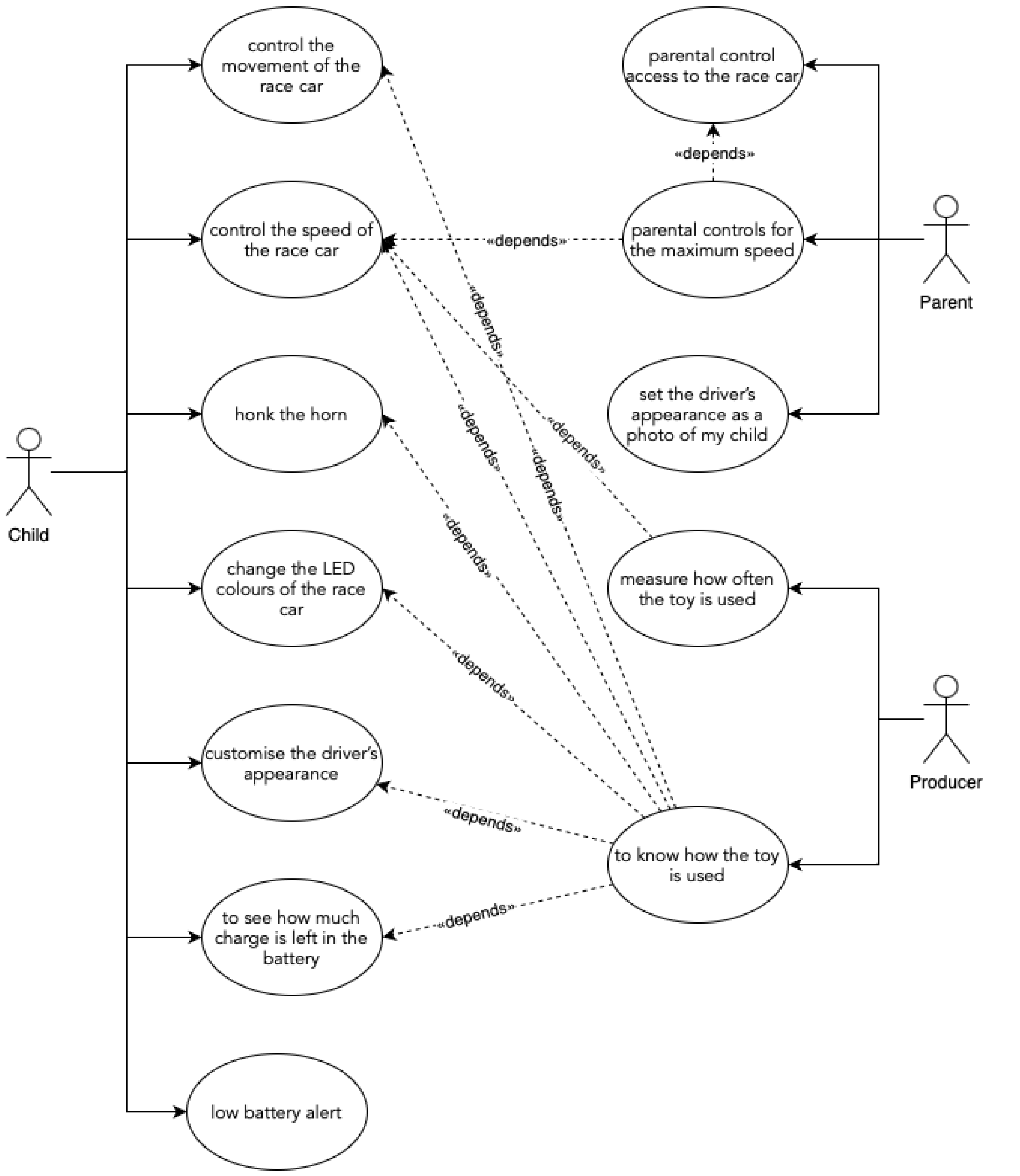
control the movement of the race car
control the speed of the race car
honk the horn
change the LED colours of the race car
customise the driver's appearance
to see how much charge is left in the battery
low battery alert
Child
parental control access to the race car
«depends»
parental controls for the maximum speed
Parent
set the driver's appearance as a photo of my child
measure how often the toy is used
Producer
to know how the toy is used
«depends»

***Figure 1**. Use case diagram.*

Design Overview

The car is central in our design. All settings, including customisation and parental control, and user analytics, will be stored in the car's local storage.

The app, working in play or parental mode, will connect via Bluetooth to issued commands. The car will exchange data with the app to use it and deliver the analytics to the remote server. Such data will be anonymised in compliance with the General Data Protection Regulation (GDPR) (Hussain *et al*., 2020).

Multiple instances of the app can control the car, but not concurrently. As per **Fig. 2**, the app can always take over in parental control mode, whilst it would use the "first-come, first-server" logic (Liu *et al*., 2014) when in play mode.

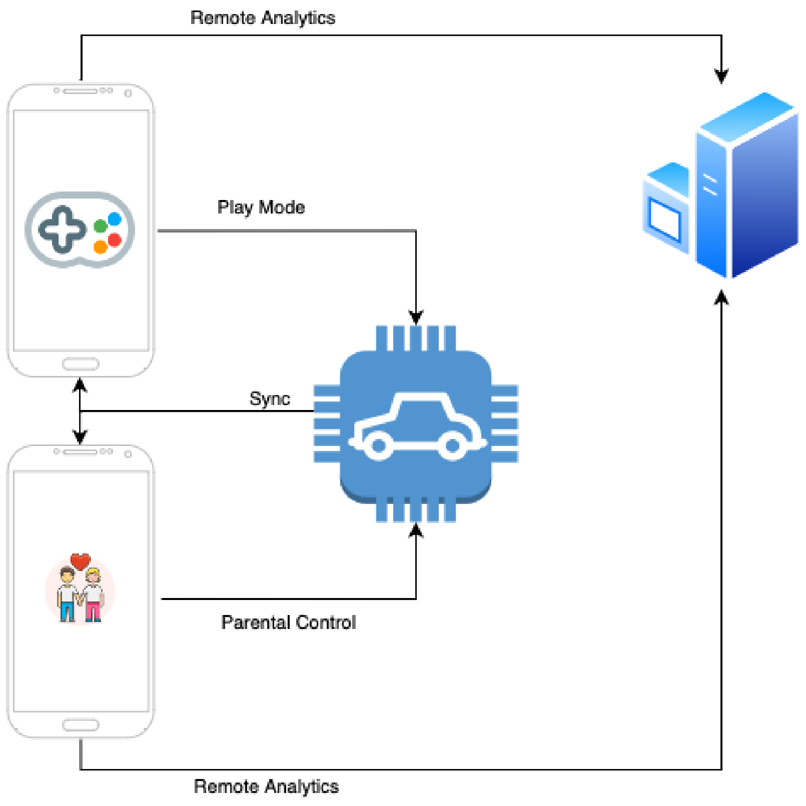

Figure 2. *High-level architecture overview.*

Project strategy

The project will be developed using Agile practices to accelerate development time, accommodate a geographically dispersed development team, and mitigate uncertainty in the requirements (Al-Saqqa et al., 2020). The Scrum methodology has been selected to deliver the application and add value incrementally by receiving and leveraging early feedback from key stakeholders.

The development strategy will make use of Behaviour-Driven Development (BDD) to link the tests to the requirements (Alferez *et al*., 2019). All requirements will be translated into Gherkin to allow for technical inspection by non-technical staff (Alferez *et al*., 2019).

Software testing will be automated via the required unit tests, but user testing of the software alone will be performed manually by a tester. A safety officer will inspect the artefacts to ensure they all meet the minimum safety requirements. A test group composed by guardians and children will provide feedback after every Agile iteration. The product will be reviewed by the Product Owner of the toy producer after every interaction. The compliance with privacy laws will be evaluated throughout and upon completion of the project to verify that anonymised data are collected and processed accordingly.

Sprint plan

The project’s timeline is illustrated via the Gantt chart in **Fig. 3**, which captures and visualises the key deliverables and the dependencies amongst them (Aslam & Ijaz, 2018; Nundlall & Nagowah, 2021) as well.

One release typically lasts three fortnightly sprints (Mahnic, 2011); considering that the Winter break at the University of Essex is from December 20th, 2022, to January 9th, 2023, the team will have 34 days to work on this software development-related project from January 10th, 2023, to February 12th, 2023.

The release goal is to devise a software to control a race car remotely, meeting at least all the must-have requirements as outlined in the section 'Requirements and Objectives'.

The following sprints' goals will fulfil the overarching release goal:

- Sprint 1:
  - Build race car and application objects and infrastructure.
  - Develop software to control movement and speed.
  - Enable it to honk the horn.
- Sprint 2:
  - Visualise the battery level and alert the user if the toy has no battery left.
  - Establish generic admin/parental control access to the race car.
  - Customise the driver's appearance.
  - Set parental controls for the maximum speed.
- Sprint 3:
  - Change the LED colours of the race car.
  - Set the driver's appearance as a photo of a guardian's child.
  - Add a process to send the analytics to the remote server.

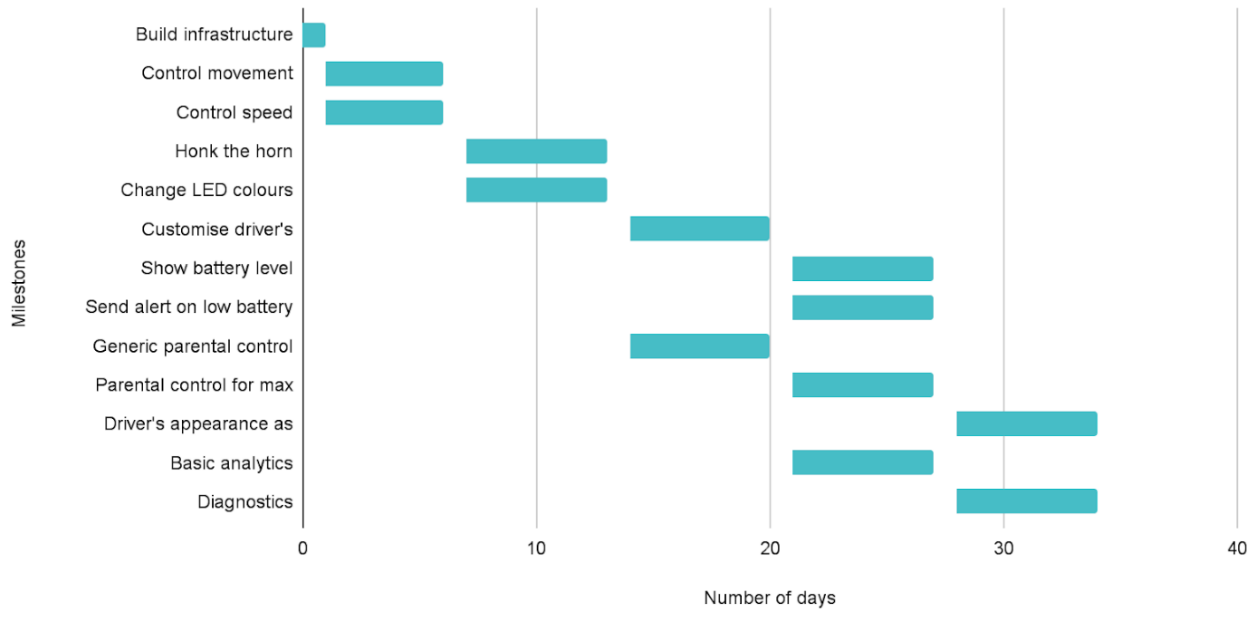

Figure 3. *Gantt chart of this software development-related project.*

Risks and Dependencies

As shown in **Table 4**, Interdependent requirements pose a risk to delivery, and mitigation strategies were explored using the Risk, Assumption, Issues, Dependencies model (IST, 2015).

Table 4*. Risks and Dependencies.*

#	Description	Severity
1	Parental control depends on the core control functionality developed at an early stage.	Low
2	Analytics depend on the development of related functions. The team will consider existing requirements to design and write code for extension.	Medium

#	Description	Severity
3	Delivery of the analytics depends on the availability of the remote server. Currently, the development team does not have access to specifications nor to a test environment.	High
4	Development team will not have access to hardware components for development and testing purposes. Thus, software may require adjustment to production hardware.	*Out of scope*

References

- Alferez, M., Pastore, F., Sabetzadeh, M., Briand, L., & Riccardi, J. R. (2019) Bridging the gap between requirements modeling and behavior-driven development. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS)* (pp. 239-249). IEEE.
- Al-Saqqa, S., Sawalha, S., & AbdelNabi, H. (2020) Agile Software Development: Methodologies and Trends. *International Journal of Interactive Mobile Technologies* 14(11).
- Aslam, W., & Ijaz, F. (2018) A quantitative framework for task allocation in distributed agile software development. *IEEE Access* 6: 15380-15390.
- Bjerke, M. B., & Renger, R. (2017) Being smart about writing SMART objectives. *Evaluation and Program Planning* 61: 125-127.
- Fine, C., & Rush, E. (2018) Why does all the girls have to buy pink stuff? The ethics and science of the gendered toy marketing debate. *Journal of Business Ethics* 149(4): 769-784. Gaikwad, V., Joeg, P., & Joshi, S. (2017) AgileRE: Agile requirements management tool. In *Proceedings of the Computational Methods in Systems and Software* (pp. 236-249). Springer, Cham.
- Hussain, F., Hussain, R., Noye, B., & Sharieh, S. (2020) Enterprise API security and GDPR compliance: Design and implementation perspective. *IT Professional* 22(5): 81-89.
- IST Information Systems & Technology Project Management Office (2015) *RAID Log*. University of Waterloo. Available from: https://uwaterloo.ca/ist-project-management-office/methodologies/project-management/planning/raid-log. [Accessed 11 Dec. 2022].

- Liu, S., Homsi, S., Fan, M., Ren, S., Quan, G., & Ren, S. (2014) Scheduling time-sensitive multi-tier services with probabilistic performance guarantee. In *2014 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS)* (pp. 736-743). IEEE.
- Mahnic, V. (2011) A capstone course on agile software development using scrum. *IEEE Transactions on Education* 55(1): 99-106.
- Nundlall, C., & Nagowah, S. D. (2021) Task allocation and coordination in distributed agile software development: a systematic review. *International Journal of Information Technology* 13(1): 321-330.
- Ogbeiwi, O. (2017) Why written objectives need to be really SMART. *British Journal of Healthcare Management* 23(7): 324-336.