Drone Control

Connor Cairns, Nisa Bayraktar, Aida Temerbulatova

May 2021

1 Overview

Faculty of Engineering in University of Bristol via Industrial Liaison Office is working in partnership with Boeing to create outreach activities. Due to the current coronavirus restrictions this presented a barrier to running in-person opportunities, the department wanted to test an online approach and potentially move to a more blended approach in the future. The purpose of these activities is to engage junior students with coding. The core of the project has to be a drone simulator. The head of the project is Steve Bullock, lecturer in University of Bristol. His area of research is dynamics and control of automated air-to-air refuelling. Currently, he is working on Drone Control project in University Flight Lab.

The aim of the project was to create an interactive platform for students aged 11-16 to learn and use basic coding concepts in an accessible format, with a longer term goal of encouraging students to engage with STEM subjects. As the client is a lecturer in aerospace engineering he wanted the project to link with the departments Flight Lab work, so to do this the students would be able to control a simulated drone on screen, the movement of the drone had to be relatively accurate in relation to how a real drone would move. To keep from throwing students in at the deep end and requiring them to write a python program for example, code blocks similar to Scratch were requested. These code blocks could be added, removed, and rearranged in a grid before running the program created on the drone.

The vision for the product solution is to create a React web app containing an interactive drone simulator that can be controlled via code blocks as mentioned previously. To achieve the realistic behaviour of the drone, an open source project, AUTONAVx, will be modified and implemented into the web app. A short introductory tutorial is available for users visiting the web page for the first time. This way students can learn to code in an interactive and accessible way. The site will not store any personal data. The progress will be saved manually at users' will. The user interface and the product functionality will be easily understandable for students aged 11-16.

2 Requirements

2.1 Stakeholders

Steve Bullock

Steve Bullock is the primary client and lead of the Drone Control project. He will be overseeing development during the project with us and maintenance or further development for the foreseeable future of the product whether that be with us or other parties.

User Stories:

1. "As the client I want the software to be well documented so that it is extendable by original and future developers."

2. "As the client, I want the web app to be accessible for my outreach students."

3. "As the client I want the web app to be visually appealing so that it attracts student's attention and provides an enjoyable coding experience."

Boeing

Boeing is not directly involved in the development progress, however, they support the project and if funding is required they will provide it. In future years if the software needs major changes or maintenance and we are not able to provide it, this may be outsourced to Boeing where they use internal staff or find external help.

ILO (Industrial Liaison Office)

The ILO is acting as an intermediary between internal clients (Steve Bullock) and external (Boeing). This could extend into the future after the project is completed if we decide to continue maintaining the project and working with Boeing while we are students.

SRAA (Student Recruitment Access and Admissions)

SRAA via the faculty engagement officer (Roisin Quinn) will be the lead on outreach with the software, running with younger students to increase interest in programming and/or Engineering. The software is aimed at users age 11-16 with the possibility of making it accessible to users from age 6.

Students(Users)

One of the users of the website will be students ages 11-16, the difference in ability that could arise due to the age gap will be accounted for and increasing accessibility is a priority for this project.

User Stories:

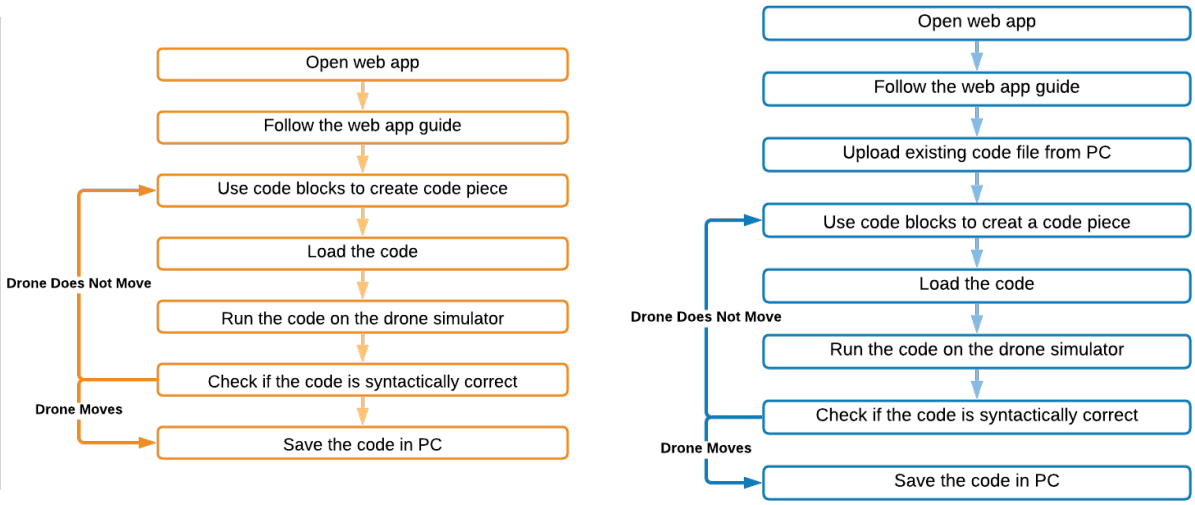
1. "As the student I want the website to be straightforward and easy to use so that I do not have to spend a lot of time learning how to use it."

2. "As the student I want engaging and challenging activities so that,they can help me to learn how to code"

3. "As the student I want to have flexible access to web app so that, I can use it anytime and anywhere"

2.2 Flow Diagrams

The flow diagrams in `figure 1` and `figure 2` are made to show how students(users) can achieve the user stories above and possible alternative and exceptional flow steps.



(a) Typical Flow

(b) Alternative Flow

Figure 1: Typical and alternative flows

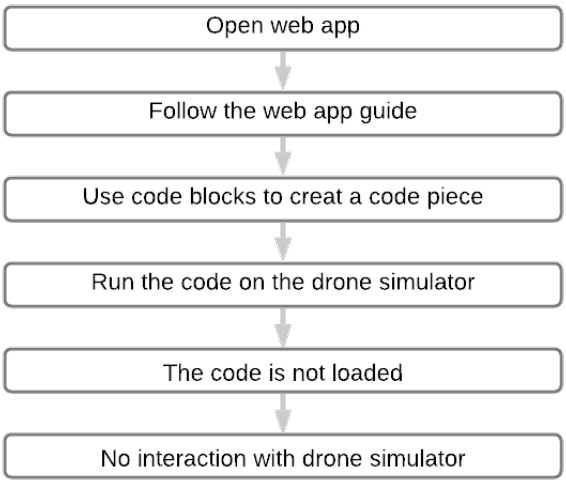


Figure 2: Exceptional Flow

2.3 Functional & Non-Functional Requirements

Building on the user stories above and a conversation with the client the following functional and non-functional requirements which can be seen in `figure 3` and `figure 4` respectively were created.

In the initial meeting with the client as mentioned, he laid out the most important requirements to him, those being:

- The software needs to be well documented so that it is extendable by original and future developers
- The website needs to support a minimum of 50 concurrent users
- The website needs to remain online and usable for a minimum of 5 years
- The students need the ability to save their programs to come back to later or share with others

	Description	Met		
		MVP	Beta	Final
1	**Have a visual drone representation**	✓		
1.1	Integrate AUTONAVx software	✓		
1.2.1	Include AUTONAVx via an iframe	✓		
1.2.1.1	Host AUTONAVx externally	✓		
1.2.2	Implement method of communication between iframe and parent window		✓	
2	**Have a wide range of code blocks to use**		✓	
2.1	Implement all AUTONAVx functionality as code blocks		✓	
2.2	Implement loop functionality			✓
3	**Support a minimum of 50 concurrent users.**	✓		
3.1	Write application in JavaScript	✓		
3.2	Serverless processing to remove potential server bottleneck	✓		
4	**Allow users to save program progress.**			✓
4.1	Allow users to download a '.drone' file			✓
4.1.1	Convert program state into JSON or similar			✓
4.2	Allow users to upload '.drone' files			✓
4.2.1	Convert '.drone' file to program state			✓
4.2.2	Update program state with uploaded program			✓
5	**Remain online and usable for 5 years.**	✓		
5.1	Obtain domain name for 5 years.	✓		
5.2	Obtain server hosting for 5 years.	✓		
5.3	Have maintainable, well documented code so that if needed, anyone is able to extend or bug fix the program.	✓		

Figure 3: Functional Requirements

	Description	Met		
		MVP	Beta	Final
1	**Must be widely accessible**	✓		
1.1	Must run on recent Chrome, Firefox and Safari. This can be achieved by using react as it supports all popular modern browsers.	✓		
1.2	Must run on Windows, Mac and Linux. As react is being used all operating systems will be supported as long as they have a web browser with JavaScript enabled.	✓		
1.3	Must be usable on less powerful systems. This can be achieved by writing efficient code and keeping dependencies to a minimum.	✓		
2	**Be accessible to users age 11-16**	✓		
2.1	Be straightforward to use and have a shallow learning curve		✓	
2.1.1	Implement a tutorial to teach new users how to use the website			✓
2.2.2	Create a self-explanatory code block experience		✓	
3	**Get students interested in studying STEM**	✓		
3.1	Gamify the program to encourage engagement			
3.1.1	Create levels with tasks of increasing difficulty for users to solve			
3.2	Have enough code blocks to allow for more advanced programs			✓

Figure 4: Non-Functional Requirements

3 Personal Data, Privacy, Security and Ethics Management

As requested by the client the Drone Control project stores no personal data. Local storage is used to store program state so users progress is not lost on refresh, this data stays on the users computer and our servers never have access to it. Although unnecessary to let users know about this there is a notice on the about page.

Ethics pre-approval was applied for on 10th November 2020.

4 Architecture

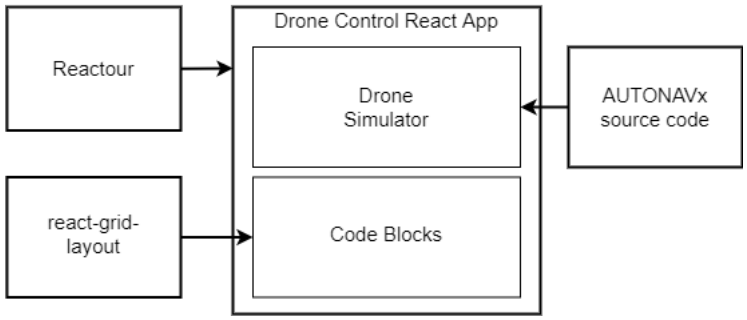


Figure 5: External Systems Diagram

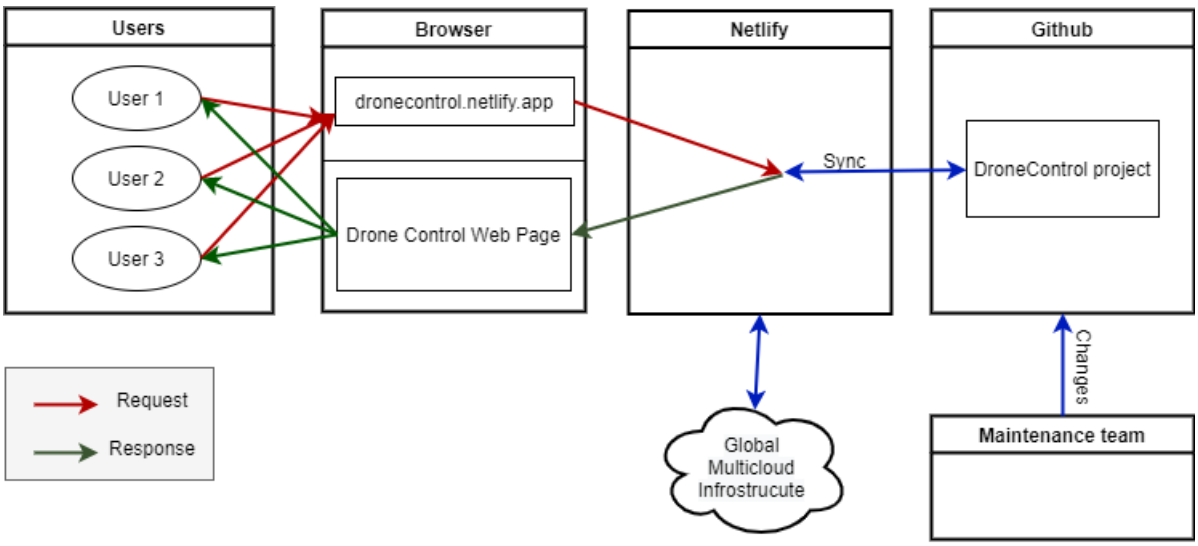


Figure 6: High Level Architecture Diagram

4.1 React App

Based on the requirements from the client it was decided to create an interactive web app using React. One of the requirements was that the web page is accessible through any browser (Google Chrome, Opera, Firefox, Microsoft Edge, etc.) and the minimum implementation should be supported on mobile devices. These could be easily achieved with React library.

4.2 Web Hosting

Another requirement was that the product had to remain online and usable for at least 5 years and should support at least 50 concurrent users. We hosted the web page using Netlify which covers these requirements. Netlify is distributed as a CDN (Content Delivery Network) but also provides additional features for managing code and assets. It deploys sites to global, multicloud infrastructure. It promises fast site delivery and high level of security for free. Also, with Netlify the web page could be easily managed at any time. The deployment process is simple and Netlify syncs with a Github repository. These features covers other two requirements from the client, namely cheap hosting of a web page and easy maintenance of the web page.

Also, the client specified that he doesn't want to save any user data. For that reason we do not need a database. All the changes to the web page would be stored in the local storage of a users' browser. Also, the client mentioned that he wants to make the usability of the product as simple as possible meaning that authorization process is not required. Saving and uploading code to save users' progress will be performed manually on the users' devices. The web app will accept the .drone file and change the layout based on its content. Conversely, it will locally create a .drone file based on the current web page layout.

4.3 External Systems

To implement a drone simulator we used an external open-source project - AUTONAVx. In AUTONAVx users can directly change source code from a browser to affect the behaviour of a simulated drone. In our project we included only the simulator itself. Also in our implementation, the source code responsible for movement of the drone is being changed using grid layout system, React-Grid-Layout. Additionally, a brief tour explaining all the components of the web page is being offered to

the users. To implement this feature Reactour component was used. The implementation scheme is shown in Figure 5.

4.4 High Level Architecture Diagram

High level architecture diagram is shown in Figure 6. Users will connect to the web page through any browser. Then the browser will send a request to server through Netlify. Netlify is in sync with Github. Then, the browser will show the content of the web page back to the users. The maintenance team will be able to make any changes to the code on Github.

5 Development Testing

During development a combination of integration and unit tests were used to give a more complete test coverage. CircleCI was used for continuous integration, it was linked to the project GitHub repository and monitored all branch changes automatically. When a change was made it would build the program and run all tests, no pull requests were merged into the main branch until it passed all tests.

GitHub workflows were used to monitor code coverage which was displayed in the README, it acted as a good indication if there was tests lacking in some areas of the project.

React testing library and Jest were used to implement the tests. This combination encourages better testing practices and aims to make tests resemble how a user would use the website, this is done by testing with DOM nodes instead of instances of React components.

5.1 Integration Testing

A great deal of the integration testing revolved around the interaction between the AUTONAVx drone simulator and the code blocks. This is an integral part of the system as a whole and its accuracy is vital as otherwise the simulator will not perform the correct commands issued by the user. It was easier to do manual integration testing for this as cross origin communication is required when sending the program to AUTONAVx. To do unit tests for this new postMessage functions would have had to have been written in AUTONAVx and the frontend specifically for this which was decided as an inefficient use of time. Instead an end-to-end integration testing style was used, programs were creating using the web app and then the drone simulator observed to confirm it moved as expected.

The integration tests can be found in Figure 7.

5.2 Unit Testing

Unit tests were used to test individual components, each component was tested to make sure it rendered without an error at a minimum, and more specific tests were implemented as necessary. Due to components being dependant on others, some components were tested through other components, e.g. the Sidebar.js was tested through Ide.js as the reducer needed to be initialised.

The unit tests in Figure 8 are for Grid.js. This component was chosen as it handles the conversion of code blocks into a program readable by AUTONAVx, it was important to ensure this was working correctly as other parts of the program, namely AUTONAVx, relied on it being accurate.

Test ID	Component	Objective	Description	Expected Result	Met
1	Simulator.js	Check AUTONAVx appears in iframe	Visually confirm AUTONAVx appears on the Drone Control home page and is interactive	AUTONAVx appears on page when loaded	✓
1.1	Simulator.js	Check loading spinner appears and waits for message from iframe it is loaded	Visually confirm loading spinner appears until drone itself is loaded	Loading spinner appears on page load, when iframe is shown the drone is loaded	✓
2	Grid.js	Check program is converted to code readable by AUTONAVx and sent to iframe	Add code blocks to program, load into AUTONAVx with 'load' button then play with the 'play' button	Drone will begin to move according to code blocks	✓
3	ItemsReducer.js	Check reducer is updated when adding code blocks	Click and drag code blocks into grid	Code blocks appear through both methods of adding meaning they are being stored in the state	✓
3.1	ItemsReducer.js	Check code blocks can be removed and this is updated in reducer	Click 'remove' button on code block already in grid	Code block is removed from grid meaning this has been updated in state	✓

Figure 7: Integration Tests

Test ID	Objective	Input	Expected Result
1	Check a basic program can be created	**items** = ["forward.1", "turn left.2", "forward.3"] **variableData** = {"forward.1": {value: 1}, "turn left.2": {value: 45}, "forward.3": {value: 1}}	**cmds** = [cmd.forward(1), cmd.turn_left(45), cmd.forward(1)]
2	Check a for loop program can be created	**items** = ["for loop.0", "forward.1", "turn left.2", "end for.3"] **variableData** = { "for loop.0": {value: "2", variable: "i"}, "forward.1": {value: 1}, "turn left.2": {value: 45} }	**cmds** = [cmd.forward(1), cmd.turn_left(45), cmd.forward(1), cmd.turn_left(45), cmd.forward(1), cmd.turn_left(45)]
2.1	Check a nested for loop program can be created	**items** = ["for loop.0", "forward.1", "for loop.2", "turn left.3", "end for.4", "end for.5"] **variableData** = {"for loop.0": {value: "3", variable: "i"}, "forward.1": {value: 2}, "turn left.3": {value: 90}, "for loop.2": {value: "2", variable: "j"} }	**cmds** = [cmd.forward(2), cmd.turn_left(90), cmd.turn_left(90), cmd.turn_left(90)...] (x3)
3	Check a while loop program can be created	**items** = ["forward.0", "while.1", "backward.2", "end while.3", "left.4", "while.5", "right.6", "end while.7", "up.8"] **variableData** = {"backward.2": {value: 1}, "forward.0": {value: 1}, "left.4": {value: 1}, "right.6": {value: 1}, "up.8": {value: 1}, "while.1": {value: "3", variable: "i"}, "while.5": {value: "3", variable: "i"} }	**cmds** = [cmd.forward(1), cmd.backward(1), cmd.backward(1), cmd.backward(1), cmd.left(1), cmd.right(1), cmd.right(1), cmd.right(1), cmd.up(1)]
3.1	Check a nested while loop program can be created	**items** = ["while.0", "forward.1", "while.2", "turn left.3", "end while.4", "end while.5] **variableData** = {"while.0": { value: "3", variable: "i"}, "forward.1": {value: 2}, "turn left.3": {value: 90}, "while.2": {value: "3", variable: "j"}}	**cmds** = [cmd.forward(2), cmd.turn_left(90), cmd.turn_left(90), cmd.turn_left(90)...] (x3)

Figure 8: Unit Tests

6 Release Testing

6.1 Core User Story

"As the client I want the web app to be visually interactive and convenient for improving programming skills so that, students can enjoy while they are creating their own code."

6.2 Testing Strategy

As the agile development strategy was being followed, throughout the project meetings were arranged with the client to get feedback on progress and ensure development was going to plan. In these meetings progress was tested by having the client act as a test user to check if the test cases (as shown in `Figure 9`) were achieved for each phase. As the test cases are mainly the requirements of the client and he is the key stakeholder who will distribute the web app, the core user story above has been chosen to obtain the most realistic results.

Phase	Test Case	Intended Outcome	Achieved
1. Minimum Viable Product	1.1 2D Drone simulation implementation	The drone simulator must be visible on the Home page of the web app	✓
	1.2 Availability of code blocks	The student must be able to choose 'Scratch'-style code blocks from the side bar	✓
	1.3 Flexible usage of code blocks	The student must be able to implement code blocks by dragging them to the coding panel and remove them from the panel by clicking the "remove" button	✓
2. Beta Release	2.1 Interaction between code blocks and drone simulator	The student must be able to see the drone movements according to the implemented code blocks	✓
	2.2 Diversity of possible drone movements	The student must be able to implement a distance value into loops and choose different type of code blocks to direct the drone.	✓
3. Final Release	3.1 Re-usable and editable existing work	The student must be able to save their code on their device by downloading as a ".drone" file and upload it back to web app	✓
	3.2 Auto-saved current state	The student must not lose their work even if they accidentally leave or refresh the page.	✓
	3.3 Interactive activities for students	The students must be able to do and complete game style coding activities.	
	3.4 Suitability of activities for different ages	The students must be able to choose their difficulty level and move to next level after they complete it.	
	3.5 Understandable layout	A new user must be able take a tour with guidance pop-up to explore the program features.	✓

Figure 9: Release Testing Table

7 UML Diagrams

7.1 High-Level Overview

`Figure 10` shows a high level overview of the Drone Control project. It outlines the major containers of the program, at the top level there is the Drone Control web app, this is what is returned to the user when they submit a http request.

The web app is made up a number of react components, the components create a component tree which can be seen in `Figure 11`, properties (props) are passed down the tree, and JSX is returned from each component. The components also rely on a reducer to provide a lot of shared state, any component underneath `Ide.js` is able to access and modify this state which will update throughout the components.

There is one major external component, AUTONAVx, this provides the drone simulator itself. It is embedded into the web app via an `iframe` and secure communication between `iframe` and parent happens using `postMessage()`.

Using an external component for the drone simulator was mainly a design decision by the client, they suggested using the AUTONAVx software, however this software had many other benefits, its licence allows for modification and distribution as long as the Drone Control project is also open source which the client was happy with, it was 'ready to go' with python commands controlling the drone, and it allowed for more of a focus on the frontend interface which given the short project period was a great help.

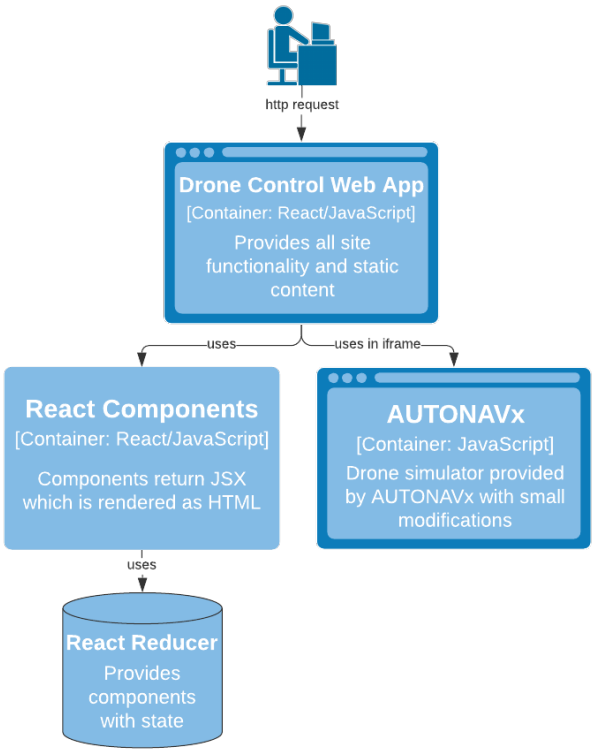


Figure 10: Container Overview Diagram

7.2 Static UML Diagram

`Figure 11` shows the static UML diagram which is made up of functional React components. Functional over class components were chosen to allow the use of hooks and to keep up with modern React, functional components are now recommended and documented better by React and as the client requested the code be maintainable for at least 5 years this was the obvious choice.

The items reducer can also be seen in the diagram, this was included as it is an integral part of the system as a whole, containing the state for the program the user is creating and providing this state to a number of components which can be seen in the diagram.

7.3 Dynamic UML Diagram

`Figure 12` shows the dynamic UML diagram for accessing the `index.html` page, it shows the path the web server takes when first loading the web app.

After a user sends a http request to the server it returns with a response of `index.html`, React handles the rendering of this and updates which components are shown as necessary.

In this example the `Home.js` component has been requested, the component first checks for any local storage, if there is none it can be inferred that this is the users first time visiting and they can be shown a tour of the website. After this the other components `Home.js` renders have to be requested, namely `Simulator.js` and `Ide.js`.

`Simulator.js` is where AUTONAVx is rendered. As mentioned previously it is embedded in an iframe, so this is requested from the URL where it is hosted and the JSX including this iframe is returned. Similar happens for `Ide.js`.

Once `Home.js` has all the components it is able to return the complete JSX and the web server can send its response.

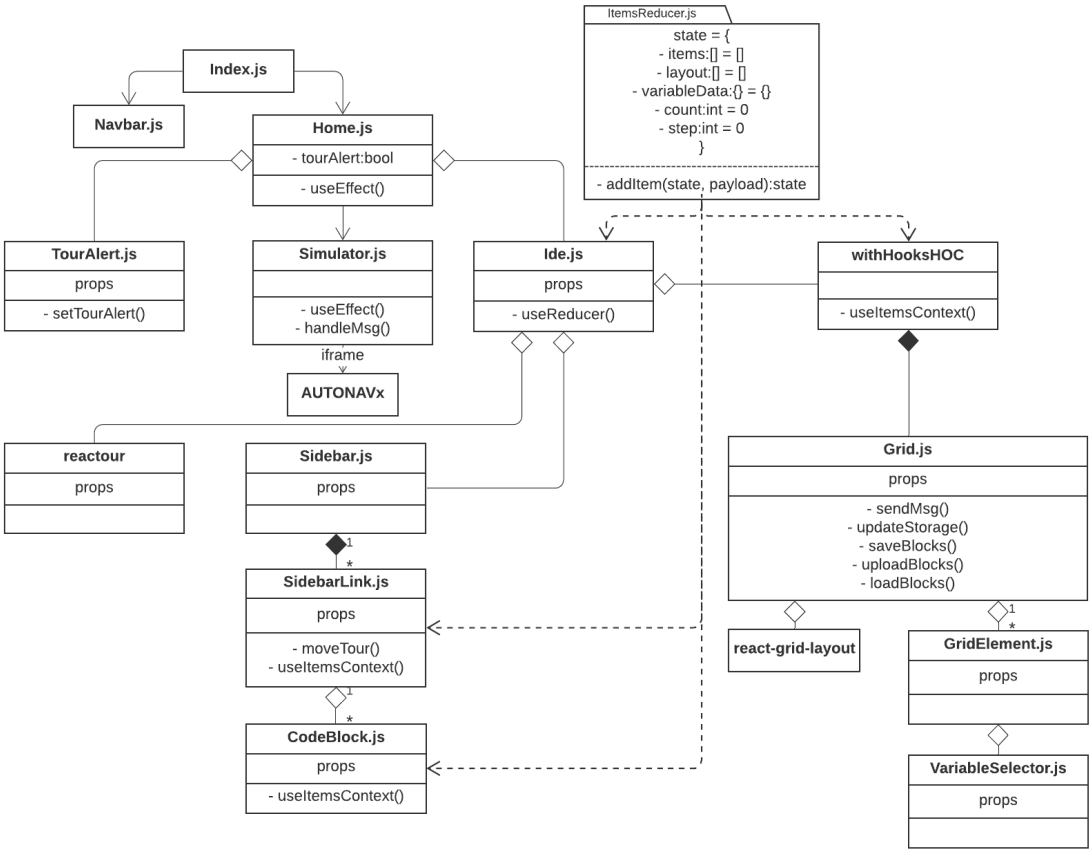


Figure 11: Static UML Diagram

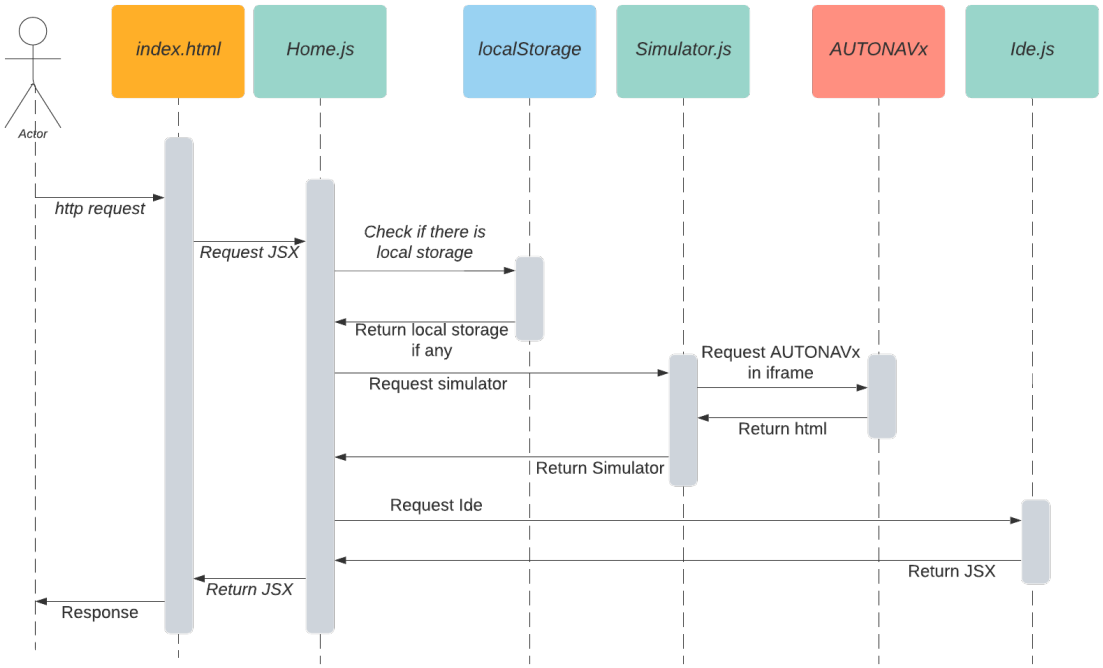


Figure 12: Dynamic UML Diagram

7.4 Reflection

The decision to use a React reducer and context to access state throughout the program was partly made to avoid 'prop drilling'. This is when you pass props through multiple components, which may or may not need that bit of state, to allow a component much further down the tree to use the state. Looking at the static UML diagram this was not as big of an issue as originally thought, state would only have had to go through a few components. However the other benefits of using a reducer can not be ignored, it allowed the state to be managed separate to any component allowing greater reusability of the components, and as the reducer manages all state changes it reduces the risk of a component updating the state incorrectly.

8 Acceptance Testing (Evaluation)

8.1 Approach

As mentioned in the release testing, throughout the project the team maintained contact with the client to get his opinions and requests. When the final version of the app was done, a meeting was arranged with him to do a final test and an interview about the final product to gain his thoughts on the development process. Due to lack of time and the challenge of gaining access to users for user testing during the COVID-19 pandemic, the product could not be tested with secondary school students who are the target users of the web app. However, in addition to the clients tests, the web app was tested after Beta and Final versions by also interviewing some of the family members aged 11 and 12 via video call. The reason why the interviewing method was chosen was to be with them when they are using it and make it easy and clear to express themselves by asking additional questions related to their feedback. The outcome from the client interview and testers feedback were crucial for the testing approach, and any insight gained allowed for improvement, even if the testers were not the exact target group. Although the user tests could not done with the 11-16 age group, in later years these tests will be done by the client who will distribute the web app, and already plans to maintain and extend it.

8.2 Feedback

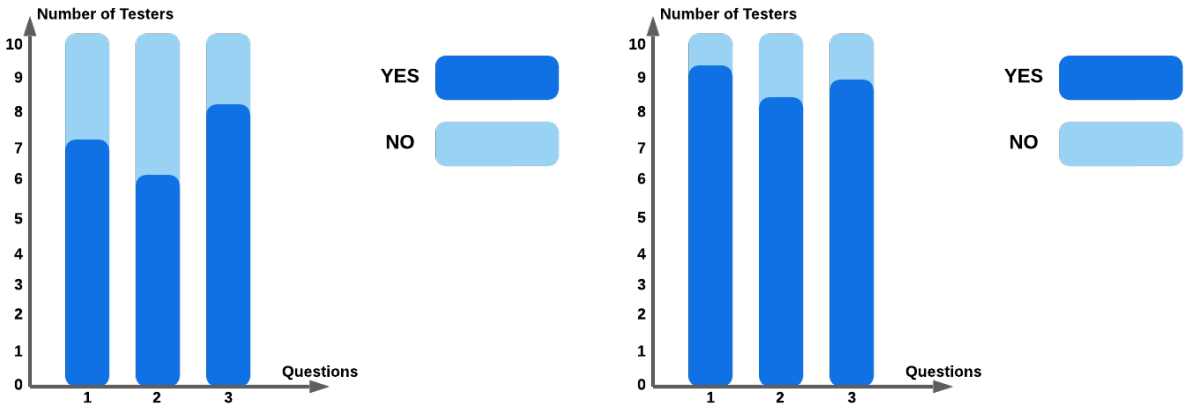
Client's Feedback

The final version of web app is suitable for student use and reached its intended goal. Even though some suggested features are missing such as coding activities for students and generating difficulty level, the project still met most of the requirements for taking it to an efficiently usable level.

Testers' Feedback

Questions that were asked during the video calls:

1. Can you understand how the web app works?
2. Can you use code blocks easily to implement your code?
3. Do you find it enjoyable and interesting to use?
4. If you had a chance what kind of features that you want to have?
5. Do you have any suggestions about the app that should be changed or improved?



(a) Testers' Beta App Feedback Chart

(b) Testers' Final App Feedback Chart

Figure 13: Beta and Final App Feed-backs

Questions	Tester's Reply
4	4.1 I want to have a short explanation about how the web app works
5	5.1 The drone movements can be shown clearer

Figure 14: Tester's Beta App Feedback Table

Questions	Tester's Reply
4	4.1 I want to have a short training demo at the start to familiarise how to use code blocks and move the drone.
	4.2 I want to have a little games or tasks to complete.
5	5.1 The distance between the blocks on the grid can be clearly defined so that, I can implement the exact distance by counting them

Figure 15: Tester's Final App Feedback Table

The feedback for beta app was discussed between the team members to improve our product for final release. New features are added such as web app guide to make the features more understandable and drone trial to show the drone movements clearly based on testers beta app feedback as shown in `Figure 14`. After getting the feedback for final product, the results changed dramatically and got better as compared in `Figure 13a` and `Figure 13b`.Therefore, user tests had a significant role in taking the web app to the next level.

By analysing the final app feedback results, the missing requirements that were mentioned by the client will be considered with development team along with testers' views as shown in `Figure 13b` and suggestions as shown in `Figure 15` to plan possible future improvements.

9 Reflection

9.1 Success of the Drone Control Project

Overall we believe the project was a success, we achieved 94% of our initial requirements including all that were related to critical site functionality meaning the web app had all necessary features to function as intended. This was confirmed in the last meeting with the client as mentioned in the acceptance testing, where he explained he was happy with the outcome. He also shared the next steps plan for the project which included future student developers extending it and integrating it into other software, at which stage they may be able to add the remaining requirements we did not have time for.

9.2 Faced Challenges and Solutions

Challenge: The team was initially a group of 5, unfortunately after a few weeks two team members could not be contacted. Therefore, the remaining three team members experienced a heavy workload throughout the project.
Solution: The workload was redistributed and new processes plan made by the team members, the requirements were also readjusted to account for a smaller team as we were not able to output the same quantity of work anymore.

Challenge: One of the technical challenges was implementing an interactive drone simulator which was one of the fundamental requirements of the web app. Creating a drone simulator is a project in itself and with a now smaller team was looking extremely difficult.
Solution: Coding a drone simulator from scratch would have been difficult given the short project period, and likely not to a high standard even if we had managed to create something. Instead the open source drone simulator AUTONAVx, suggested by the client, was adapted to our needs and new features added based on the requirements of integrating this.

Challenge: The client initially forgot to mention he wanted a serverless web app so it would be possibly to host it for cheap or better, free. We found this out after the backend creation had already started. Therefore, it caused a time loss and a change of process plan.
Solution: A meeting was arranged between team members to make a new process plan by discussing the possible ways about how to build a serverless web app. An agreement on letting the user download a .drone file to their computer was reached and we went forward with this decision.

9.3 Success of the Processes

There is an argument to be made that as overall the project was a success, that processes chosen must have also been a success. While this is true to an extent and certain processes of the project

worked very well, namely our interview style testing which highlighted some areas that were lacking such as UI simplicity, some processes likely could have been improved.

More time spent planning and designing the project before we began would have likely saved us time in the long run. By doing this we could have presented the idea of including a backend to our client before any work started, this was mentioned as one of our biggest challenges due to the time lost which could have been avoided.

9.4 Impact

As stated above the project met all requirements needed to function, meaning it can be deployed and be used by students as was planned. The students will be able to gain an understanding of basic coding concepts by controlling a drone in a simulator with easy-to-use code blocks. We hope the project attains its goal of encouraging more students to study STEM subjects and believe there is a good chance this will happen, especially with the plans of the client to continue to work on the project to improve and extend it so that it can be online and working for many years. We believe the project still has a lot of potential past this year.

There are no negative ethical implications of the project, there is no user data stored and no user tracking so no potential for a leak of user information. The project is also not disruptive technology, it is unlikely Scratch will be going out of business due to this.