**Technical Plan**

Our application consists of a two-player Top Trumps like the card game in which each player has a deck of cards, each representing an animal with its traits and statistics. During a turn, a player will choose a card and an attribute and set the card down; the opposing player will then choose a card and place it down. The card with the highest value in the chosen trait will win the ‘battle’, as the virtual models of the animals chosen are augmented on top of the cards and interact with one another.

The use of augmented reality is ideal for a simple, interactive card game to enhance the user experience for both players as well as present opportunities to support autistic users. Using a simple two-dimensional surface, such as the face of a card, makes identifying and projecting augmented reality models much simpler than targeting three-dimensional objects.

A feature of the game will include the ability for players to upgrade and unlock new cards using points gained from playing games. The number of points earned during a game will scale, for each consecutive day two players play one another, the number of points they gain will increase. A player will also receive a bonus for playing against a new opponent for the first time. We chose to implement this with this system of offering point to promote regular interaction between players as well as to encourage playing with new players, with the goal of helping an autistic player to meet new people and create friendships by playing with others regularly.

An example scenario in which an autistic user would benefit from such a game would be an autistic child who finds it difficult to talk to and interact directly with others. During a game, they can place their card and choose an action through the user interface without the need to talk out loud. An option to send pre-written messages and emojis to the other player can help them to bridge the communication gap without the need to interact with them directly.

The user interface will use an android device to replicate a Microsoft HoloLens, to allow the user to interact with the augmented reality aspect of the project. The project will largely interact with events and commands that happen in reality and translate these into a function to happen within the application.

The user interfaces itself will be simple, offering a projection of the real-world on-screen, in the user’s view. The application will constantly be checking to recognises any real-world objects around it, such as analysing the distance and size of surfaces and patterns, to identify a card that forms part of the play-space. This will then augment a three-dimensional model and the cards traits and statistics into the view. For example, seeing a card on a surface will result in the application projecting what appears to be a three-dimensional model of the card on to the surface, using both the surface size, distance and angle from the user to decide the scale of the model.

The application will have a small head up display projected on top of the camera view. This will allow the user to see things such as card information, points, current players turn and allow the user to access the menu options. Through here they can change various settings such as text sizes and colours.

The HUD will occupy the perimeter of the view. For example, points for each play will appear across the top, while a menu button appears at the bottom. Keeping track of the turn will be central, at the top of the screen, so the user can always see the current state of the game. While the display the real-world, in the event of a recognised object being identified the display will project a model visible only through the augmented reality application, this model is likely to be in the centre of the players view, with a list of actions and card statistics alongside it.

Interaction with the application will be done in two ways. First will be the analysing of the real world. This is achieved simply by looking around the real-world play-space; this is an automated process in which the application will constantly analyse to recognise surrounding objects. Through this process, the application will be able to determine what information to display as well as where and how it should be displayed. Complex calculations in the background will take place – particularly on calculating the angle and draw-distance of the augmented models. All of this happens with no user input other than looking around with the camera.

Secondly will be the HUD commands, in which the user can gesture with their hands given the camera to click and select actions and options. The application will recognise the user’s gestures and the position of the user’s hand to determine the action to take. This will allow players to interact with the game to choose a cards to attribute to play, zoom in and out to view card information, send messages to the other player and access the main menu of the system.

Alternatively, the HoloLens provides support for voice commands. This would enable the user to use their voice to perform direct interaction with the application. For example, if the application detects the word “*Menu*”, it could then bring up the menu, removing the need for hand gestures which could prove challenging for players with autism. Allowing either hand-gestures or voice commands will greatly enhance the accessibility of the application to a wider range of players

We will be producing the project in cycles and iterations, generating an overall vision of the project and using many software engineering methodologies such as Agile and SCRUM to produce the project to a high standard. The prototype methodology is an ideal methodology as we are using new technology, Augmented Reality, that our team members have not had much experience in developing.

Prototyping means producing a high-quality playable build for the product but not producing a complete version of the product we are making. We will be listening to close feedback and results from testing to develop further and improve the application.

Since we are aiming our software at people with Autism, their feedback is most important. We will be aiming to interview and gain feedback from people who more closely fit our target audience to produce a play space better to suit their needs.

Using this prototype approach allows for a more structured and clearer idea of how we should approach the development process, as we can see what works and what does not work. We can utilise player statistics and information gathering software and market research to supplement the development and testing process to understand our users better and tailor the software for them.

Using GitHub to store and version control our code helps us to easily raise issues and assign tasks and identify the latest features and flag up bugs during testing. This will make the development and testing process easier, works well with Agile methodologies and is the industry standard for developing software.

Along with a Kanban board and Microsoft Teams, GitHub will help us keep a well-organised project structure, direct communication with other team members with the use of issue tracking, comments and versions control. Since we will be using the Unity Game Engine to develop our application with multiple developers contributing, the version control and team management feature that GitHub provide will be beneficial.

We will be sharing code on GitHub; each team member will be working on a feature to get working. Each task will be marked on a developer’s to-do list with a deadline. We will have a guideline of coding conventions that we must follow to produce working, connected snippets of code which is readable and reusable. Each developer will be working on a feature and will update their progress regularly with stand-up meeting and issues posted on GitHub’s issue tracking system.

Also, we will regularly discuss if things should be implemented, taken out or left as is and decide when a new version is ready to be tested and released. We will use different branches to separate versions which are ready for testing or release. This separates development builds and release builds, to keep untested and potentially volatile code from tested, working code.

The solution that we have agreed upon as the basis of our project is designed to assist children on the autism spectrum with challenges faced during social interaction, particularly with other children during play. We have decided to face this issue by designing a simple card game similar to Top Trumps. The game will involve physical cards that have can be scanned and recognised by the smartphone or HoloLens device to project a three-dimensional model using augmented reality.

The feasibility of the implementation of this augmented reality technology is high, as using a scannable code or image is less technically challenging than detecting a physical surface such as a table or floor, or a three-dimensional object, ensuring it is within the scope of a team of students who have no experience with augmented reality.

The design and implementation of an appropriate user interface will bring up more feasibility issues, due to the difficulties autistic children can face with understanding and perceiving different types of information. Applications that focus on augmented reality, particularly games, can have problems with the user interface not correctly integrating with the physical space, for example, clipping through surfaces or being hard to distinguish against different backgrounds. Due to this, there will be a focus on ensuring that these user interface problems are managed.

Finished project files will be saved and respected in a common GitHub repository, allowing for group viewing and task reviews from team developers while maintaining tested and released code. GitHub will be regularly updated with the new versions of our working software as well as the individual tasks submitted by group members.

Tasks that have already been completed will be available for reference; future jobs will be available to view for developers to research and plan around. These tasks will be equally distributed between developers while taking the developers proficiencies into account to minimise differences in workloads and improve efficiency.

Timeboxing will be used to ensure work is completed in time for deadlines and is not rushed to produce high-quality work. Regular meetings will be held at the end of each timebox to allow developers to discuss how they have done in each task and if any problems or changes need to be implemented.

When working on a task, developers must follow the design specification, if changes are required for ease of implementation or because of time restraints all group members must be made aware of these changes and a discussion should take place before they are finalised.

If group members have questions between timeboxes, they should make use of our team chat and GitHub repository to ask them.

As well as the master GitHub branch, we will maintain other branches for testing. This process comes directly after the task is completed by the developer after being tested and approved; it can be merged with the master branch.

If a task requires a significant amount of time but is difficult to break into smaller chunks for each developer to work on, the task may be assigned to more than one member, in which case is maintaining common source code, and continuous communication is essential.

**Significant risks and actions to avoid, reduce or tackle them**

|  |  |
| --- | --- |
| Risks | Actions |
| Disagreement on core aspects of the application resulting in wasted development time. | Frequent discussions and meetings to ensure every member of the team consistently agree on what the core aspects should be. |
| Poor productivity due to time between deadlines. | Timeboxes will be short and frequent to provide the team with a sense of urgency and ensure that every team member’s contribution is accounted for. |
| The scope of the project slowly increases until it is unrealistic to accomplish within the given timeframe. | Robust documentation of the scope of the project that is clear for every team member. Critical analysis of any proposed expansions or major changes to the project. |
| The software has major bugs or unknown issues when close to the final deadline. | Every commit to the GitHub repository will be reviewed by at least two other members to ensure there is minimal bugs/issues. Adequate time will be allocated for testing and bug fixing. |
| Significant pieces of work are lost/destroyed due to external factors, e.g. hard drive corruption. | All work done by the team will be frequently saved and backed up to multiple places such as GitHub and Google Drive. |

**Estimation of costs**

£10 per hour for each developer, based on the wage of an entry-level software developer role.

Our development team consists of 6 developers, with each developer contributing roughly 50 hours of development time to the project:

*Cost per developer = 50(hours) x £10(per hour) = £500  
Total development team cost = £500(per developer) x 6(developers) = £3,000*

We will be using Unity Personal because our team will be making less than $100,000 and are therefore permitted to use Unity Personal for free. We will also only be using free Unity store assets which will incur no extra cost.

We will be using the Microsoft Office 365 Business package this will cost £7.90 per developer, per month. This is also an annual commitment, costing £94.80 per year per developer.

*Microsoft Office 365 team package = £94.80(per developer) x 6(developers) = £568.80*

The total cost for the development of the project will be £3,568.80 (£3,000 in developer costs and £568.80 in software licensing).