# User Stories

| # | User Story | Independent | Negotiable | Valuable | Estimable | Small | Testable |
|---|---|---|---|---|---|---|---|
| | | *Is the user story self-contained?* | *It is not an explicit contract for features* | *How is it valuable to customers?* | *The time required for this can be approximated* | *The story can be fulfilled as an atomic item* | *How can this be tested (or if it's a functionality how can the customer try it out)?* |
| 1 | As a **Player**, I want to **be able to see every tile and/or the buildings on it** so that I can **choose how to make my move.** | The ability to see the game tiles and buildings is a standalone feature that does not depend on other functionalities, like movement mechanics or turn-based logic. | The story states that players need to see all tiles but does not dictate how this is provided, allowing the development team flexibility in implementation. | Players desire to see the board when playing which can assist in their decision making and provide visual feedback. | Clear scope, minimal unknowns. The team can confidently assign story points after clarifying. | Purely observational (no gameplay logic). Placeholder art acceptable for the first iteration. | When a solution is produced, the customer may be able to interact with it. |
| 2 | As a **Player**, I want to **see when I have won or lost** so that I can **know when the game is over.** | The act of displaying a win/loss state does not depend on how the game determines a winner. The logic for deciding if a player has won or lost can exist separately, and this story only focuses on presenting that result to the player. | The story specifies the need for players to know when the game is over, but it does not prescribe how this information is conveyed. | Visual feedback is important for users as it may be frustrating otherwise. | The effort to check win/loss conditions and show a message can be reasonably estimated. | This represents only a small part of the win/loss logic, specifically the display of a message, making it a small and manageable user story. | Trial users can be asked to play a game and note to an observer if they know if the game has finished and who has won. |
| 3 | As a **Beginner Player**, I want **a clear indication of whose turn** it is so that **I can make my moves without confusion.** | The turn indicator is just a UI element that updates based on turn changes, which can be displayed without requiring movement or actions to be completed first. Thus, it is independent of other game mechanics. | Defines the clear need for a turn indicator without suggesting how it can be implemented. | If the user mistakenly thinks that it is their turn, it may cause confusion and dissatisfaction if they are locked out of acting. | The effort to implement a turn indicator can be estimated based on the UI design. | It focuses on a single, specific feature: indicating the active player. | Trial users can be made to communicate to the observers if at any point the users are confused. |
| 4 | As a **Player**, I want to **know which phase – either build or move – of the game I am in** so that I can **know what actions I can be taking.** | This story only requires visually indicating which phase the player is in that does not necessarily depend on the mechanics of building or moving. | This is negotiable because it defines the need to show the game's phase without specifying how it should be presented, allowing flexibility in implementation to enhance the player's experience. | Users (especially those that are new) may struggle to recall the rules and may become confused as to what available actions there are. | The effort to show the current phase is predictable based on UI requirements. | Focuses solely on communicating the current phase, not implementing the phase logic. | Trial users can be made to communicate to the observers if at any point the users are confused. |

| # | User Story | Independent | Negotiable | Valuable | Estimable | Small | Testable |
|---|---|---|---|---|---|---|---|
| | | *Is the user story self-contained?* | *It is not an explicit contract for features* | *How is it valuable to customers?* | *The time required for this can be approximated* | *The story can be fulfilled as an atomic item* | *How can this be tested (or if it's a functionality how can the customer try it out)?* |
| 5 | As a **Beginner Player**, I want to **access the game rules easily** so that **I can understand how to play and follow the correct gameplay mechanics.** | This user story is independent because the ability to access the game rules does not rely on other features being completed first like the player turn or other game mechanics. | This is negotiable as it specifies that the rules must be made accessible for the user, but allows the developers to implement it without any restrictions. | Users may struggle to remember the rules for which the game should provide a way to learn. | The effort to provide access to rules like linking a file can be estimated. | It focuses only on making the rules accessible. | Trail users can be told to find the rule/book and rate how readable it is. |
| 6 | As an **Opponent**, I want **the ability to build another story on top of my opponents building** so that I can **restrict their movement**. | This user story is independent because the ability to build on top of an opponent's structure does not rely on other mechanics. Even if movement is not fully developed, the action of stacking buildings can still be implemented separately, ensuring that this feature can be worked on in isolation. | This is negotiable as it covers an important game mechanic of building whilst ensuring that it doesn't describe how it is being built. | Encourages strategic play by allowing players to restrict their opponent's movement, making the game more competitive and engaging. | The logic for allowing/disallowing such builds can be scoped and estimated. | It's a focused piece of functionality: enabling one specific action during gameplay. | Trial users can attempt to build on their opponent's structures.<br><br>**Test Case:**<br>Given an opponent's worker is on a Level 1 space, when the player builds a Level 3 structure on an adjacent Level 2 space, then the build should be permitted.<br><br>**Expected Result:**<br>The building height increases, restricting the opponent's movement. |
| 7 | As a **Player**, I want to **move my worker to an adjacent space following the game's movement rules**, so that I **can progress toward building and winning**. | The ability to move to the correct building story is a fundamental mechanic that can be developed separately from other features like building structures or win conditions. Even if additional mechanics are not yet implemented, the core movement logic can still function and be implemented on its own. | This is negotiable as it focuses on the needs of the player, which is moving legally, and allows flexibility in how movement rules are implemented. | Maintains game balance by preventing unfair advantages and promotes longer, more strategic gameplay. | Movement logic based on adjacency and rules can be clearly scoped and estimated. | It focuses on just one action of moving a worker according to the rules. | Trial users can attempt to move their character up two stories in a single move to verify if the game allows it.<br><br>**Test Case:**<br>Given a worker is on a Level 1 space, when they attempt to move directly to a Level 3 space, then the move should be denied.<br><br>**Expected Result:**<br>The move is blocked, and the worker remains in their original position. |

| # | User Story | Independent | Negotiable | Valuable | Estimable | Small | Testable |
|---|---|---|---|---|---|---|---|
| | | *Is the user story self-contained?* | *It is not an explicit contract for features* | *How is it valuable to customers?* | *The time required for this can be approximated* | *The story can be fulfilled as an atomic item* | *How can this be tested (or if it's a functionality how can the customer try it out)?* |
| 8 | As a **Beginner Player**, I want **a tutorial** so that I can **learn how to play the game**. | Similar to game rule access, the implementation of a game tutorial can be done independently and does not depend on other game mechanics | Clearly targets new users and focuses on the why and not the implementation. Hence, it doesn't specify if the tutorial is interactive, text-based or a video. | A tutorial helps new players understand the game mechanics quickly, reducing frustration and making the game more accessible. | The scope of creating a tutorial can be outlined and estimated based on the chosen approach. | If scoped properly (e.g., covering only the basics), it's a focused and manageable task. | Trial users can be asked to check if they can manually start the tutorial.<br><br>**Test Case:**<br>Given a player is in the main menu, they should be able to select a tutorial option that starts the tutorial.<br><br>**Expected Result:**<br>The tutorial begins as expected. |
| 9 | As a **Player**, I want **the game to respond quickly to my actions** so that **gameplay feels fluid.** | Performance optimizations, such as reducing lag or improving frame rates, can be worked on regardless of whether specific game features (like movement or building) are fully implemented. It focuses on improving the game's responsiveness without being directly tied to a particular functionality. | This is negotiable as it focuses on user perception and lets the team discuss solutions such as optimising animations or reducing delay in other ways. | Performance is often overlooked but is especially important for games as the users for this sector are much more particular in regards to latency. | Performance improvements can estimated easily, especially for key interactions | Focuses on responsiveness, which can be targeted through specific optimizations. | The game should be benchmarked when run, ensuring that the performance when profiled stays consistent throughout. |
| 10 | As a **Beginner Player**, I want to **confirm my move before playing** so that **I can avoid misclicks and ensure my actions are correct.** | Even if other gameplay elements are still being developed, a confirmation prompt can function on its own, allowing players to review and adjust their moves before finalizing them. Thus it can be implemented independently. | This is negotiable as it addresses a player's desire for control and does not specify how the confirmation will occur, whether its a button or a timeout. | Confirming moves before finalizing ensures players feel confident, prevents mistakes, and enhances the overall gaming experience. | The time required to implement a confirmation step can be clearly estimated. | It focuses only on confirming an action, not changing the game logic itself | Trial users can be asked to observe if they feel if they noticed that this feature exists and if it felt intuitive to use. |

| # | User Story | Independent | Negotiable | Valuable | Estimable | Small | Testable |
|---|---|---|---|---|---|---|---|
| | | *Is the user story self-contained?* | *It is not an explicit contract for features* | *How is it valuable to customers?* | *The time required for this can be approximated* | *The story can be fulfilled as an atomic item* | *How can this be tested (or if it's a functionality how can the customer try it out)?* |
| 11 | As a **Player**, I want to **have a standalone executable** so that I can **play offline and with only one computer.** | Creating a standalone executable focuses solely on packaging the game for offline play on a single computer, which can be developed and tested separately from other game mechanics like in-game features. | The story is negotiable because it doesn't specify how the executable is built (e.g., installer vs. portable) or whether it includes DRM. The team can discuss trade-offs like dependencies, packaging tools, or offline feature scope. | Users may not have a stable internet connection when playing. Additionally this would mean that a server doesn't not need to be run (say for a web-app) only for distribution | The task of packaging the game as a standalone executable is manageable and can be estimated based on platform requirements. | Creating the executable is separate from core gameplay or complex features. | Ensure that a standalone executable can be built. |
| 12 | As a **Player**, I want to **build in available positions** so that I can **start constructing the tower and set up for future builds**. | The fundamental feature of placing buildings in available positions can be developed and tested independently. | Doesn't specify how "available positions" are shown (highlighting, blocking invalid clicks) or tower visuals. | Enables players to progressively build towers, which is essential for winning the game. | The complexity of checking available positions and enabling building can be estimated based on the game's layout. | The task is focused on allowing construction in available spaces, not on complex building or tower logic. | Given a player moves their worker to an adjacent space, when they attempt to build on an adjacent empty space, then the game should allow the build and update the board. |
| 13 | As a **Player**, I want to **place both workers on valid starting spaces during setup**, so I can **strategically position them for the game**. | This focuses on the initial setup of the game which can be implemented and tested independently, before the rest of the game features (like turn order or victory conditions) are fully developed. | Doesn't specify how positions are chosen (click-to-place, drag-and-drop, or pre-defined zones).<br><br>Allows the team to define placement rules (e.g., workers must be on the ground level, spaced apart, or on the board's perimeter). | Gives players control over their starting positions, allowing for strategic placement from the beginning of the game. | The task to validate and place workers on the board is clear and can be estimated easily. | It's a focused task on placing workers during the setup phase, not impacting the entire game. | Given a new game has started, when the player selects two valid empty spaces and places their workers, then the workers should be positioned accordingly on the board. |
| 14 | As a **Player**, I want to **be able to choose a god card at the start of the game** so that I can **have a special power that influences my strategy during the game**. | Even if the mechanics involving how the god card affects gameplay like special powers are not fully developed, the feature of selecting a god card at the start of the game can be implemented independently. | Doesn't specify how god cards are chosen (random draft, pick-ban, UI layout).<br><br>Leaves room to define power balance (e.g., beginner-friendly vs. advanced gods). | Allows players to customise their gameplay experience and develop unique strategies based on their chosen god power. | The time required to implement card selection and its effects can be estimated based on the game's existing framework. | It's a focused feature, enabling god card selection at the start of the game, only to be considered at the start of the game. | Given the game setup phase, when the player selects a valid god card from the available options, then the chosen god card should be assigned to them and displayed on the screen. |

| # | User Story | Independent | Negotiable | Valuable | Estimable | Small | Testable |
|---|---|---|---|---|---|---|---|
| | | *Is the user story self-contained?* | *It is not an explicit contract for features* | *How is it valuable to customers?* | *The time required for this can be approximated* | *The story can be fulfilled as an atomic item* | *How can this be tested (or if it's a functionality how can the customer try it out)?* |
| 15 | As a **Competitive Player**, I want to **see a timer that limits my turn time to 1 minute** so that the **game progresses quickly and remains competitive**. | The timer only affects the pacing of the game and does not depend on other features, allowing it to be worked on and tested in isolation. | Doesn't mandate how the timer is displayed (visual bar, countdown clock, sound warning at 10s).<br>Allows debate on timer rules (pauseable? adjustable time limits? grace periods?). | Ensures that turns do not drag on for too long, keeping the pace of the game fast and engaging, particularly for competitive players. | The time and effort required to integrate a timer and its effects on turn mechanics can be scoped and estimated. | It's a focused feature that only adds the timer and associated turn limits, without complicating other gameplay aspects. | This can be tested by checking if a timer appears when a player's turn is ongoing and that the player's turn is over once the timer runs out. |
| 16 | As a **Player**, I want a **settings menu before gameplay to customize audio, controls, and visuals**, so I can **optimize my play environment**. | The ability to adjust game settings, such as sound, can be implemented separately from other gameplay mechanics like movement, building, or player interaction. | Doesn't specify which settings (sound/music volume, keybinds, graphics, rulesets) or UI design (sliders, toggles).<br><br>Allows the team to prioritize settings (e.g., basic vs. advanced options). | Provides personalisation options, allowing players to tailor the game to their preferences, which can improve their overall experience. | Involves setting up a settings menu which is an estimable task. | It revolves around overall game settings and has no correlation with the actual game mechanics, making it small. | This can be tested by looking for a settings option in the main menu and clicking on it to see if it takes you to a settings screen. |
| 17 | As a **vision impaired player**, I want to **be able to change themes** so that **the game is more accessible for me.** | This focuses on customizing the game environment to accommodate vision-impaired players, which can be done regardless of other gameplay features being fully implemented. The theme adjustment can be worked on and tested independently. | Doesn't specify exact colors or UI changes (flexible for team to research best practices).<br><br>Allows implementation as presets or adjustable sliders (e.g. brightness, saturation). | Enhances accessibility by allowing visually impaired players to adjust colours and contrasts for better readability and usability. | The work required to implement accessible themes like high contrast or color adjustments can be estimated. | It focuses only on theme customization for accessibility | Given a player accesses the settings menu, when they select a different theme, then the game's visuals should update accordingly to reflect the chosen theme. |
| 18 | As a **Competitive Player**, I want a **post-game summary with relevant match data (like build efficiency or win/loss)**, so I can **track my strategy over time**. | This feature is only triggered after a match ends, meaning it does not interfere with the game's primary functions. Even if other mechanics are still in development, the end-of-match screen can be designed, tested, and refined independently. | Doesn't specify which stats to show (e.g., win/loss, turn count, tower height) or layout (minimal vs. detailed).<br><br>Allows the team to decide if stats are shareable/savable (e.g., screenshots, leaderboards). | A random end of game message which points out an outstanding statistic can be interesting to the player improving engagement and encouraging different decisions. | The effort required to gather, display, and format the match data in the post-game summary can be estimated. | It's a focused feature that adds a post-game report without impacting other aspects of gameplay. | A set of trial users could be given a set of match screens and have them rank which message they found interesting/prefered. |

| # | User Story | Independent | Negotiable | Valuable | Estimable | Small | Testable |
|---|---|---|---|---|---|---|---|
| | | *Is the user story self-contained?* | *It is not an explicit contract for features* | *How is it valuable to customers?* | *The time required for this can be approximated* | *The story can be fulfilled as an atomic item* | *How can this be tested (or if it's a functionality how can the customer try it out)?* |
| 19 | As a **Player**, I want to **personalize my workers with distinct colors/accessories,** ensuring they **remain visually distinct from opponents' workers**. | The customization of the character's appearance and accessories can be developed separately from core gameplay mechanics like movement, building, or scoring. | Doesn't specify how customization works (pre-set skins, color picker, unlockables).<br><br>Allows the team to define scope (cosmetic-only vs. gameplay-impacting items). | Customizing a character's appearance allows players to express their individuality, fostering a deeper connection to the game and enhancing their overall enjoyment. | Separate from other gameplay mechanics and revolves around character customization for which time required can be estimated. | It's a focused task on customizing visuals for workers, without affecting core gameplay or mechanics. | This can be tested by checking if a player's workers can be accessorised and they testing if the changes are saved and can then be viewed on the main board. |
| 20 | As a **Competitive Player**, I want to **view my total number of wins and losses** so that **I can track my performance over time.** | It relies on storing and displaying match results, which can function independently of other features. Even if the game rules or mechanics change, the system can still record and present a player's win/loss record without affecting gameplay. | Aspects like data presentation, time range, update frequency, and additional metrics can be adjusted based on user needs and technical feasibility. | It's valuable to a competitive player to see their performance over time, as it gives them feedback and adds motivation. | The effort required to track and display win/loss data is manageable and can be estimated. | The task is limited to implementing the win/loss tracker and its display, | It is testable as you can verify that the wins and losses are tracked correctly and displayed in the right place. |
| 21 | As a **Beginner Player**, I want to **receive a message if I try to make an invalid move**, so that **I know I need to choose another action.** | The validation of moves and error messaging can be implemented separately from other gameplay mechanics | Doesn't specify how the message is delivered (pop-up text, sound effect, UI shake).<br><br>Allows the team to define tone (error sound vs. subtle highlight) and detail (generic "Invalid!" vs. rule-specific hints like "Can't climb 2 levels"). | This provides value because it guides players, ensuring they understand when they are making invalid moves, which improves the gameplay experience. | The task of checking for valid moves and showing appropriate messages is clear and can be estimated. | This feature focuses specifically on move validation and feedback, without affecting other gameplay systems. | You can test if invalid moves trigger the correct message, and that the message is displayed in a readable format. |
| 22 | As a **Player**, I want a **'Rematch' button after a match ends**, so I can **replay with the same settings or return to setup**. | The functionality of being able to restart a game can be implemented independently as It simply resets the game state and does not depend on any other feature. | Clarifies flexibility (same settings vs. fresh config).<br><br>Keeps UI/flow open for discussion. | This is valuable because it improves user experience, allowing players to keep playing without needing to manually reset or close the game. | Defined behavior (reset board while preserving settings) and clear UI requirement (button placement) make effort predictable. | Focused solely on post-game reset logic. Complexities like stats preservation or new setup configurations can be added later. | It's testable because you can verify that the game restarts correctly after the player chooses the option. |

| # | User Story | Independent | Negotiable | Valuable | Estimable | Small | Testable |
|---|---|---|---|---|---|---|---|
| | | *Is the user story self-contained?* | *It is not an explicit contract for features* | *How is it valuable to customers?* | *The time required for this can be approximated* | *The story can be fulfilled as an atomic item* | *How can this be tested (or if it's a functionality how can the customer try it out)?* |
| 23 | As a **Player**, I want **the game to end once a player's worker has reached the third story** so that **I know the game is complete.** | This feature does not rely on other parts of the game being fully developed, and it can be tested independently to ensure the game ends when the condition is met. | Doesn't specify how the win is communicated (pop-up, animation, sound fanfare).<br><br>Allows the team to decide if the game allows post-win actions (e.g., undo for misclicks) or ends instantly. | This provides clear value to the player by defining a winning condition, ensuring the game has a proper ending. | Clear victory conditions (worker on L3) and defined termination behavior make effort predictable. | Focused solely on win detection. Celebration sequences/leaderboard updates can be separate stories. | This can be tested by checking if the game ends when any player's worker is placed on the third story. |