AGILE REFERENCE STORIES

Story #	Story Pool	Story Source	Story Summary
1	5	EECS 268 - Lab 4 (WebBrowser)	I'd rate this lab at five story points. I have to build a linked list from scratch and then use it to simulate a browser's navigation history. Implementing insert, remove, and traversal logic is time-consuming, and correctly handling back, forward, and navigate commands requires careful state management and detailed testing to ensure everything works smoothly.
2	2	EECS 268 - Lab 5 (Recursion)	I'd rate this lab at two story points. I need to write three separate recursive programs—a power calculator, a flu outbreak model based on the previous three days, and a Fibonacci solver with two modes. Each requires careful recursion design and base cases, plus user input validation and formatted output, making it moderately challenging but manageable.
3	1	EECS 168 (Lab 1)	I'd rate this lab at one story point because it's a simple, introductory task with very low complexity. The work mainly involves setting up GitHub, writing basic Python print statements, and submitting files correctly. There's no algorithmic problem-solving or debugging required—just following clear directions and completing straightforward steps.
4	3	EECS 330 - Hashmap (Lab 7)	I'd rate this lab at around three story points. It involves moderate complexity since I need to implement and test several connected methods for a hash map, handle edge cases like duplicates and removals, and add logic to find the maximum passengers. It's manageable but requires careful coding and debugging.
5	2	EECS 468 - Assignment 7	This assignment involves writing and testing several Haskell functions with both debug and test cases, following strict formatting and documentation requirements. It also requires detailed comments, collaboration disclosure, and proper submission structure. I'd rate it two story points because it was fast to create but still complex and demands precision, organization, and thorough documentation.
6	1	EECS 388- Homework 1	I'd estimate this homework at about one story point. It's straightforward and mostly involves applying what I already knew about binary conversions, endianness, and basic C concepts. There's no real debugging or creative coding required—just careful attention to detail and accurate calculations, so the effort and complexity are low.
7	13	EECS 210 - Assignment 5	This assignment asks me to implement a function–relation analyzer and a recursive Sudoku solver with backtracking across five puzzles, produce formatted outputs, screenshots, and exhaustive comments, and package everything correctly. I'd rate it thirteen story points for multi-part scope, recursion complexity, I/O, testing, and strict documentation requirements.
8	5	EECS 330 - Lab 9 (Graph Traversals)	I'd rate Lab-9 at five story points. I need to implement BFS/DFS shortest paths (handling "no path"), count connected components, and a bipartite check across possibly disconnected graphs, then validate on multiple test graphs and package results for GitHub. Multiple algorithms, careful correctness, and thorough testing make it moderately complex and time-consuming.
9	5	EECS 330 - Lab 4 (Linked List)	This two-part lab has me implementing a deque twice (doubly linked list and NumPy array with dynamic resize), completing a full API, writing palindrome and reversal utilities, plus Part B helpers (wordToDeque, OffByOne, OffByN) with tests and packaging. I'd rate it five story points for multi-implementation scope, pointer/resize edge cases, and thorough validation.

AGILE REFERENCE STORIES

Story #	Story Pool	Story Source	Story Summary
10	8	EECS 678 - Lab 6 (Dining Philisopher's)	I'd rate this lab at eight story points because it requires modifying multithreaded C code to implement two synchronization-based solutions to the dining philosophers problem. It involves complex reasoning about mutexes, condition variables, and deadlock prevention, demanding careful debugging, testing, and understanding of concurrency control principles.
11	8	EECS 678 - Lab 11 (/proc filesystem)	I'd rate this lab at eight story points because it builds on prior multithreading work and adds system-level interaction through the Linux /proc filesystem. It requires understanding process and thread structures, reading and parsing kernel data using file operations, and implementing deadlock detection logic—demanding careful reasoning, debugging, and precision in C.
12	13	EECS 678 - Buddy Allocator	I'd rate this at thirteen story points. Implementing buddy_alloc/buddy_free with correct split, placement, and recursive coalescing across orders, precise free-list bookkeeping, and buddy math (bitwise XOR) is systems-level tricky. Passing strict simulator I/O, crafting edge-case tests, and debugging fragmentation/off-by-one bugs adds significant effort and risk.
13	2	EECS 168 - Lab 5 (Grocery Log & Pirate)	I'd rate this lab at two story points. It's mostly straightforward list/string practice: menu-driven grocery list, neighbor lookup via indices, and careful input handling. Complexity rises slightly for EECS 169 with the string shifter; that variant feels like five story points due to added transformations.
14	3	EECS 168 - Lab 3 (Loops)	I'd rate this lab at three story points because it focuses on practicing while and for loops, string and range operations, and basic input/output handling. The exercises—number guessing, ASCII conversions, and mathematical calculations—are straightforward and require logical thinking but minimal complexity or debugging effort.
15	13	EECS 678 - Scheduler lab	I'd rate this lab at thirteen story points. Implementing a multi-core scheduler across FCFS/SJF/PSJF/PRI/PPRI/RR, plus a working priority queue, requires careful preemption logic, tie-breaking, quantum handling, and correct averages (waiting/turnaround/response). Matching simulator timing/order rules and passing 54 reference tests adds significant integration, debugging, and edge-case effort.
16	1	EECS 658 - Assignment 1	I'd rate this at one story point. It's moderate: environment setup and library installs, a simple CheckVersions script, then an NBClassifier with 2-fold CV on Iris, reporting accuracy, confusion matrix, and P/R/F1. Manual metric calculations, screenshots, and strict packaging add effort, but the modeling task itself is straightforward.
17	8	EECS 658 - Assignment 2	I'd rate this at eight story points. I need to implement 2-fold CV, wire up seven models, and print labeled confusion matrices and accuracies for a 150-sample test set. The work is moderate: data splits, consistent pipelines for linear/polynomial regressors vs classifiers, result aggregation, screenshots, and clean packaging/documentation.
18	3	EECS 268 - BoardGames Lab	I'd rate this lab at three story points. It requires designing multiple classes (Executive, BoardGame), robust file I/O for TSV data, building and maintaining a list of objects, and implementing several menu-driven queries with validation and edits. The work is moderate: object modeling, sorting/filtering logic, clean user interaction, and readable formatted output with sensible class responsibilities.

Requirements Stack

ID	Requirement Description	Story Points	Priority	Sprint Number
R1	Implement RESTful FastAPI backend to serve sports data (teams, players, stats, odds) using PostgreSQL and Redis caching.	13	1	1
R2	Integrate live data feeds from APIs for football, basketball, baseball, and UFC. Normalize and store in unified schema.	13	1	1
R3	Build WebSocket interface for real-time game and stat updates to connected clients.	8	1	1
R4	Create front-end interface (React Native or PWA) to display teams, players, and stats.	8	1	1
R5	Implement search and filtering by sport, team, or player across datasets.	3	1	1
R6	Implement user-defined alerts for news mentions and injury updates of subscribed teams.	8	2	2
R7	Implement scheduled background jobs (APScheduler) to poll for new data, injuries, and odds updates.	5	2	2
R8	Integrate betting odds API and display odds for upcoming events with bookmaker info.	5	2	2
R9	Store and display recent injury reports per team/player.	3	2	2
R10	Optimize DB queries and introduce Redis caching for high-traffic endpoints.	3	3	3
R11	Add automated tests (unit, integration) for backend services.	5	3	3
R12	Containerize services with Docker and deploy via Docker Compose.	3	3	3
R13	Add mobile notifications for live game updates and subscribed team news.	5	4	4
R14	Create admin dashboard to manage API keys, data sources, and logs.	5	4	4
R15	Implement data deduplication and validation pipeline for ingest jobs.	5	4	4
R16	Design responsive UI supporting both desktop and mobile clients.	5	4	5
R17	Add authentication (JWT) for user accounts and subscriptions to teams/players.	5	5	5
R18	Build Discord bot/webhook integration for sending alerts and live updates.	3	5	5

Requirements Stack

ID	Requirement Description	Story Points	Priority	Sprint Number
R19	Ensure compliance with betting data policies and user age verification.	2	5	5
R20	Add analytics module tracking user engagement and alert metrics.	3	5	5