BidNow - Sprint Review Document

A real-time auction platform where registered users can bid on listed items, track ongoing auctions, and manage their listings. Admins can manage users, listings, and track metrics.

Sprint Name: Sprint 3 – Componentization, Testing, and Code Quality
Sprint Goal: Build functional components, write tests, and implement build optimizations.
Duration: 16 hours.
Team Project: BidNow – Online Auction System
Pod Members:

- Geethu Joseph
- Govindh RM
- Jayashree VS
- Karthika RS
- Kushagra Srivastava
- Mayukh Paul.

Submission Date: 4 July, 2025

❖ What Was Planned vs. What Was Delivered

- **User Stories for each member task:**

Member 1 (Geethu Joseph): Build auction card component with bid button.

- As a user, I want to view each auction in a card format so that I can easily browse and interact with listings.
- As a bidder, I want a bid button on each card so I can quickly place a bid.
- As a developer, I want to modularize the auction card as a reusable component to improve code reusability.

Member 2 (Govindh RM): Create bidding form and manage state.

- As a user, I want a form to enter bid amount so I can participate in auctions.
- As a developer, I want to manage input form state using hooks/reactive forms so that validation and updates are handled properly.
- As a QA engineer, I want to ensure invalid inputs are flagged before submission.

Member 3 (Jayashree VS): Setup and enforce ESLint rules.

- As a developer, I want to follow consistent coding standards using ESLint so that the codebase remains clean and maintainable.
- As a team lead, I want to prevent common syntax errors by enforcing strict linting rules.
- As a team member, I want to be notified in my IDE/terminal if I break any rule.

Member 4 (Karthika RS): Add unit tests for auction list and bidding form.

- As a QA engineer, I want to write unit tests to verify the behavior of the auction list and bidding form.
- As a developer, I want to mock API responses and test dynamic UI changes using Jest.
- As a user, I want confidence that features work correctly even after updates.

Member 5 (Kushagra Srivastava): Configure Webpack for optimized bundling.

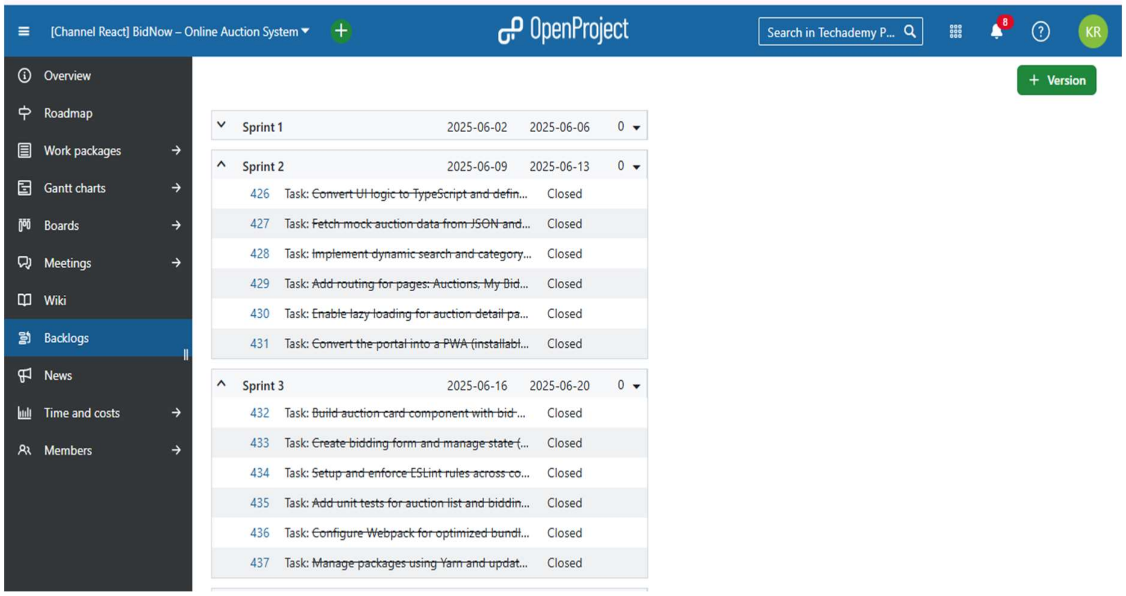
- As a developer, I want the app to load faster by optimizing bundle size via Webpack.
- As a build engineer, I want to split code using dynamic imports so only necessary parts are loaded.
- As a user, I want the initial load to be smooth and fast.

Member 6 (Mayukh Paul): Manage packages using Yarn and update README.

- As a team member, I want to manage all dependencies using Yarn so versions remain consistent.
- As a new developer, I want the README to include setup steps and scripts to run the app easily.
- As a contributor, I want to understand how to install, build, and test the app quickly.

Planned Task	**Status**	**Delivered**
Auction Card Component with bid button	Completed	Modular Auction Card created with dynamic data rendering and bid functionality
Bidding Form & State Handling	Completed	Controlled form with validation and React state management
ESLint Configuration	Completed	Configured ESLint with recommended rules, Prettier plugin, and custom overrides
Unit Tests for Auction List & Bidding Form.	Completed	Used Jest and React Testing Library to test key UI logic and component behavior
Webpack Optimization	Completed	Configured code splitting, minification, and lazy-loading with optimized Webpack config
Yarn Package Management & README	Completed	Yarn used for consistent dependency management; README includes setup and run instructions

❖ **OpenProject sprint status screenshot:**



Sprint 1		2025-06-02	2025-06-06	0
Sprint 2		2025-06-09	2025-06-13	0
426	Task: ~~Convert UI logic to TypeScript and defin...~~	Closed		
427	Task: ~~Fetch mock auction data from JSON and...~~	Closed		
428	Task: ~~Implement dynamic search and category...~~	Closed		
429	Task: ~~Add routing for pages: Auctions, My Bid...~~	Closed		
430	Task: ~~Enable lazy loading for auction detail pa...~~	Closed		
431	Task: ~~Convert the portal into a PWA (installabl...~~	Closed		
Sprint 3		2025-06-16	2025-06-20	0
432	Task: ~~Build auction card component with bid-...~~	Closed		
433	Task: ~~Create bidding form and manage state (...~~	Closed		
434	Task: ~~Setup and enforce ESLint rules across co...~~	Closed		
435	Task: ~~Add unit tests for auction list and biddin...~~	Closed		
436	Task: ~~Configure Webpack for optimized bundl...~~	Closed		
437	Task: ~~Manage packages using Yarn and updat...~~	Closed		

All tasks assigned to each group member for this sprint in OpenProject were completed and closed on time.

❖ **Challenges Faced**

1. **Auction Card Component (Member 1 – Geethu Joseph)**
   - Translating legacy inline JavaScript behavior into React logic.
   - Prop drilling became an issue when integrating into larger pages (to be solved in next sprint).
   - Ensuring the timer resets correctly when props change.

2. **Form Validation Edge Cases (Member 2 – Govindh RM)**
   - Bid amounts below the minimum or above the user's current balance weren't initially blocked; introduced conditional disabling, validation logic, and error states.
   - Integrating the existing HTML form into a controlled React component was tricky, especially ensuring validation errors updated in real-time.

3. **Linting Rule Conflicts (Member 3 – Jayashree VS)**
   - Some ESLint and Prettier rules (e.g., arrow parens, semicolons, trailing commas) conflicted, leading to formatting inconsistencies and extra debugging effort.
   - Setting up .eslintrc to work across different editors while keeping plugin versions in sync was time-consuming.

4. **Mocking Complexity in Tests (Member 4 – Karthika RS)**
   - Ensuring onBidSuccess and onClose were called with the right values after form submission required careful prop mocking and event simulation.
   - React component logic tied to useEffect and state updates made it hard to write isolated tests, especially for event-driven UI updates.

5. **Webpack Bundle Debugging (Member 5 – Kushagra Srivastava)**
   - Webpack's tree-shaking optimization broke certain imports due to inconsistent use of default and named exports in reused modules.
   - Converting static HTML pages into a modular, component-based structure required rewriting the import/export logic and configuring loaders properly.

6. **Yarn Workspace Conflicts (Member 6 – Mayukh Paul)**
   - Faced a lot of issues when converting older Bootstrap HTML pages into component-based structure due to inline styling and tight HTML/JS coupling.
   - Encountered confusion when integrating Yarn with pre-installed global npm setups across member systems, which was later solved by standardizing instructions.

❖ **Learnings**

1. **Componentization:**
   - Learned to break UI into isolated, testable, and reusable parts using props and state effectively.
2. **Testing Practices:**
   - Gained experience with unit testing tools, mocking, and code coverage metrics to improve test reliability.
3. **Linting Discipline:**
   - ESLint helped enforce readable, consistent code and flagged issues during development instead of at runtime.
4. **Webpack Usage:**
   - Understood bundling strategies like code-splitting, tree-shaking, and caching for performance boosts.
5. **Package Consistency:**
   - Realized the importance of locking package versions across team members using Yarn and yarn.lock.

❖ **Deliverables Summary**

1. AuctionCard and AuctionList implemented as reusable components using React/Angular.
2. Bidding form with validation and controlled state.
3. ESLint with Prettier plugin configured across codebase.
4. Unit test suites for auction list rendering, bid state changes, and form validations.
5. Webpack optimized for production build with minimized, files were bundled and stored in dist folder.
6. Yarn used for dependency management, with updated README and scripts.
7. Additionally, Admin Dashboard implemented alongside User Dashboard, enabling admin users to manage registered users, view auction item listings, and monitor bidding activity.

Github link to repository: https://github.com/geethu26/BidNow/tree/sprint-3
The above github repository consists of the entire codebase in React in the bid-now directory. A Readme file has also been provided with setup instructions and other details of the sprint.

- **Conclusion:**
Building on the interactivity and dynamic data flow established in Sprint 2, Sprint 3 marked a significant transition for the BidNow platform — from static page logic to a fully componentized React application. One of the major milestones was the conversion of traditional HTML and CSS files into modular, reusable React components, which significantly improved code structure and maintainability. The team adopted a component-driven development approach, enforced ESLint rules for consistent coding practices, and introduced unit testing to ensure UI behavior was reliable and verifiable. Webpack was configured for performance-optimized bundling, and Yarn was used to manage dependencies efficiently. With a clean, scalable, and tested React codebase now in place, BidNow is technically equipped to move forward with implement APIs, bidding logic, authentication, and DB connectivity features in the upcoming sprint.