**Document 1: Definition of Done**

**Definition of Done (DoD) for EWL in Estate Account**

The **Enterprise Work List (EWL)** in the **Estate account** typically involves managing and delivering a portfolio of tasks related to enterprise-scale estate management systems or processes. These tasks could include various aspects like property management, financial transactions, user story implementation, or system upgrades.

In this context, applying a robust **Definition of Done (DoD)** is critical for ensuring that all work meets the agreed-upon standards, aligns with customer expectations, and is deployable to the production environment without issues. Here’s a detailed breakdown of each **DoD** point:

**1. Produced code for presumed functionalities**

* **Context for EWL in Estate Account:** This refers to ensuring that the functionality described in the user story (such as implementing a new feature to handle property listings, financial calculations, or tenant management) is fully coded and integrated.
* **Details:** The development team must produce the code for the anticipated functionalities without cutting corners. This includes the design of all relevant modules and interfaces within the estate management system. For example, if the task involves improving a property search feature, all aspects of the code responsible for filtering, sorting, and displaying property data must be written and functional.
* **Why it’s important:** Ensures that the functionality outlined in the project backlog is fully developed and can be tested as part of the feature delivery process.

**2. Assumptions of User Story met**

* **Context for EWL in Estate Account:** The assumptions (such as specific requirements around user roles, permissions, and system constraints) related to the user story need to be checked and validated.
* **Details:** Each user story comes with assumptions (e.g., certain user roles have access to particular functionalities or a feature must work on specific browsers). These assumptions should be met as per the agreed-upon scope and should be reviewed before declaring the user story as "done."
* **Why it’s important:** Ensures that the functionality is built according to the initial assumptions and is aligned with user expectations.

**3. Project builds without errors**

* **Context for EWL in Estate Account:** This refers to ensuring that the development work can be compiled and built without encountering issues in the continuous integration (CI) or deployment pipeline.
* **Details:** The code should be free of build errors when compiled and should pass through automated build processes without issues. For example, if you are working on an estate account system that integrates with external APIs for property data, it should compile successfully in both development and test environments.
* **Why it’s important:** A clean build ensures that the feature is stable enough for testing and that no blockers exist before deployment.

**4. Unit tests written and passing**

* **Context for EWL in Estate Account:** This includes writing unit tests for each component of the system (e.g., property calculation engine, user authentication, payment system).
* **Details:** The code should include unit tests that cover critical functions and business logic, and all tests should pass successfully. For instance, if a user story involves a system for tracking rent payments, unit tests should verify that payments are correctly recorded and categorized.
* **Why it’s important:** Unit tests help ensure that individual parts of the application function correctly, minimizing the risk of defects and regressions.

**5. Project deployed on the test environment identical to the production platform**

* **Context for EWL in Estate Account:** The feature or functionality must be deployed in a test environment that mirrors the production environment closely to ensure compatibility.
* **Details:** For an Estate account system, this could include deploying new property management features or updates to a staging environment that matches the production environment (e.g., same database, application server, and third-party integrations). This helps catch issues related to environment-specific configurations or dependencies.
* **Why it’s important:** The test environment should reflect the production setup so that any potential issues can be identified early.

**6. Tests on devices/browsers listed in the project assumptions passed**

* **Context for EWL in Estate Account:** If the project involves web or mobile features for the estate system, this step involves testing across the devices or browsers that are critical to the system’s user base.
* **Details:** For example, if the user story involves displaying property listings, the feature should be tested on all supported browsers (e.g., Chrome, Safari, Firefox) and devices (e.g., desktop, mobile). It ensures that end users can access and interact with the system across the expected platforms.
* **Why it’s important:** Ensures a seamless user experience across all supported devices and browsers.

**7. Feature ok-ed by UX designer**

* **Context for EWL in Estate Account:** User experience is crucial for estate management applications because the system is often used by different stakeholders (property managers, tenants, buyers).
* **Details:** The UX designer should review the feature to ensure it aligns with the design principles and meets usability standards. For example, ensuring that property detail pages are clear and easy to navigate or that payment systems are intuitive for users.
* **Why it’s important:** User experience (UX) approval ensures that the feature is usable and aligns with the expectations of the users.

**8. QA performed & issues resolved**

* **Context for EWL in Estate Account:** Quality assurance (QA) testing must be performed on the developed feature to verify its functionality and ensure there are no bugs or defects.
* **Details:** QA testing should cover both functional testing (e.g., does the rent calculation feature work correctly?) and non-functional testing (e.g., performance, security). If issues are found, they should be resolved before the feature is considered complete.
* **Why it’s important:** QA ensures that the feature is free from defects and meets both functional and non-functional requirements.

**9. Feature is tested against acceptance criteria**

* **Context for EWL in Estate Account:** The feature should be tested according to the **acceptance criteria** defined at the beginning of the user story to ensure it meets all the required customer and business requirements.
* **Details:** For example, if the user story defines an acceptance criterion such as “a tenant should be able to view their payment history,” testing should confirm that this is implemented correctly.
* **Why it’s important:** Acceptance criteria define the business value of the feature and ensure that the work meets the customer’s needs.

**10. Feature ok-ed by Product Owner**

* **Context for EWL in Estate Account:** The product owner is responsible for ensuring that the feature aligns with the product vision and goals.
* **Details:** The Product Owner reviews and signs off on the feature to confirm that it delivers the intended value and meets the stakeholder’s expectations. This step ensures that the feature is not only complete but also aligned with the overall business objectives.
* **Why it’s important:** Product Owner approval ensures that the feature is viable and satisfies business requirements.

**11. Refactoring completed**

* **Context for EWL in Estate Account:** Refactoring is the process of improving code structure without changing its behaviour. This step ensures that the code is maintainable, scalable, and easier to extend in the future.
* **Details:** The code should be optimized and refactored to ensure readability, performance, and maintainability. For example, simplifying complex queries in the property management system or optimizing code for faster user interactions.
* **Why it’s important:** Refactoring reduces technical debt, making the system easier to maintain and extend in the future.

**12. Any configuration or build changes documented**

* **Context for EWL in Estate Account:** If any changes are made to system configurations (e.g., database schemas, build processes), they must be documented for future reference.
* **Details:** Any modifications to infrastructure, configuration files, or the build process should be documented clearly to ensure that the team can replicate or troubleshoot configurations as needed.
* **Why it’s important:** Proper documentation ensures transparency and makes it easier for other team members to understand or manage these changes.

**13. Documentation updated**

* **Context for EWL in Estate Account:** Documentation should be updated to reflect any changes made to the system, ensuring both technical and user-facing documents are accurate.
* **Details:** This includes updating system architecture diagrams, user guides, API documentation, and internal developer documentation to reflect the new feature or changes.
* **Why it’s important:** Well-maintained documentation is crucial for future team members, external collaborators, and end users who rely on accurate, up-to-date information.

**14. Peer Code Review performed**

* **Context for EWL in Estate Account:** Peer code reviews are essential for maintaining code quality and ensuring that the code meets the team's standards.
* **Details:** A peer code review should be conducted where another developer examines the code for issues such as bugs, coding best practices, security vulnerabilities, and performance optimizations.
* **Why it’s important:** Peer reviews ensure higher code quality, knowledge sharing among team members, and better collaboration.

**Document 2- Product Vision**

**Answer:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Scrum Project Name** | Enterprise Work List (EWL) for Estate Account | |  |  |
| **Venue** | Rajiv Gandhi, Sahyadri park Phase 3, Pune | |  |  |
| **Date:** 03/03/2025 | **Start Time:** 09:00 am | | **End Time:** 06:00 pm | **Duration:** 9 hrs |
| **Client** | John Deol | |  |  |
| **Stakeholder List** | **Rekha M** | | **Venkatesh J** | **Nishi Desai** |
|  | **Esha Jain** | | **Snehal C** | Siddhi J |
| **Scrum Team** | | | | |
| **Scrum Master**: | | **Ankur Agarwal** | | |
| **Product Owner**: | | **Satya Kadam** | | |
| **Scrum Developer 1**: | | Rita S | | |
| **Scrum Developer 2**: | | Sagar Jain | | |
| **Scrum Developer 3**: | | Sunil Pawar | | |
| **Scrum Developer 4**: | | Akhil Jaju | | |
| **Scrum Developer 5**: | | Sonal Chavan | | |

**Vision**: The vision for the **EWL for Estate Account** is to streamline estate management processes, enabling property managers, maintenance teams, tenants, and other stakeholders to efficiently manage and track all tasks, work orders, and responsibilities related to property management and operations.

|  |  |  |  |
| --- | --- | --- | --- |
| **Vision** | | | |
| **Target group** | **Needs** | **Product** | **Value** |
| **Market Segment:** Property Management Organizations, Real Estate Agencies, and Corporate Estates  **Target Users and Customers:**  **Property Managers**: Handle work orders, schedule maintenance, and track tasks.  **Maintenance Teams**: Manage tasks, track work, and communicate effectively.  **Tenants and Residents**: Submit and track maintenance requests.  **Landlords and Investors**: Track property performance and maintenance status.  **Real Estate Agencies**: Use the system for tracking properties and operations. | **Problem:**  Estate management is burdened with fragmented systems, unclear communication, and delayed tasks.  Tasks may be missed, delayed, or go untracked, causing inefficiency and poor customer satisfaction.  **Benefit:**  The **EWL for Estate Account** centralizes all work orders, maintenance requests, and property-related tasks into one platform, enabling better tracking and communication.  Automation of workflows, real-time updates, and better task prioritization improve overall efficiency and satisfaction. | **What is it?**  A **comprehensive estate management tool** that helps manage work orders, maintenance requests, and tasks related to estate operations.  **What makes it desirable and special?**  **- Centralized Management**: Single source of truth for all tasks, work orders, and updates  - **User-Centric Design**: Easy-to-use interface for all stakeholders.  - **Automated Workflows**: Streamlines common tasks and notifications.  - **Mobile and Cross-Platform Accessibility**: Available on both desktop and mobile platforms.  - **Real-Time Tracking and Reporting**: Detailed insights and updates for informed decision-making.  **Is it feasible to develop the product?**  Yes, developing the **EWL for Estate Account** is **feasible** with expertise in web and mobile development, workflow automation, and integrations with third-party systems.  **Feasibility Details:**  This includes work on user interfaces, database management, security, and ensuring proper scalability. | **Business Goals:**  - **Increased Efficiency**: Streamline tasks and reduce manual work.  - **Cost Savings**: Reduce operational costs through improved task management.  - **Improved Customer Satisfaction**: Quicker response times and better communication for tenants and managers.  - **Enhanced Reporting and Insights**: Provides detailed insights for decision-making.  **Business Model:**  - **Subscription-based SaaS Model**: Offering different subscription tiers (e.g., Basic, Premium).  - **Premium Features**: Additional functionalities available for a higher tier, such as advanced reporting or integrations.  - **Customizable Add-ons**: Offering extra features like branded portals or extra storage. |

**Document 3: User stories**

**Answer:**

|  |  |  |  |
| --- | --- | --- | --- |
| User Story No: 1 | Tasks: 2 | | Priority: Highest |
| **As a Property Manager,**  I want to be able to register a new property in the system  so that I can start tracking tasks and maintenance requests for that property. | | | |
| BV: 500 | | CP: 02 | |
| ACCEPTANCE CRITERIA:  Property registration screen with text boxes for **Property Name, Address, Owner Contact, Type of Property,** and **Date of Registration.**  Click on **Register** button.  Send **Success** notification to the Property Manager | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| User Story No: 2 | Tasks: 3 | | Priority: High |
| **As a Maintenance Technician**,  I want to view all open work orders  so that I can prioritize tasks based on urgency | | | |
| BV: 400 | | CP: 01 | |
| ACCEPTANCE CRITERIA:  Display a **list of open work orders** with **task details, property information,** and **priority level.** Option to **sort and filter** work orders by urgency, priority, or property.  **Click on a work order** to view detailed information and update status. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| User Story No: 3 | Tasks: 4 | | Priority: Medium |
| **As a Tenant**,  I want to submit maintenance requests through a portal  so that I can report issues and track progress. | | | |
| BV: 300 | | CP: 03 | |
| ACCEPTANCE CRITERIA:  Provide a **form for maintenance requests** including fields for **Issue Description, Location, Preferred Time,** and **Priority.**  **Submit request** button to send the request to property management.  **Track progress**: Tenant should receive notifications for status updates (e.g., "In Progress", "Completed"). | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| User Story No: 4 | Tasks: 5 | | Priority: Medium |
| **As a Property Owner**,  I want to receive notifications about property maintenance and work order status  so that I can stay informed about ongoing issues. | | | |
| BV: 200 | | CP: 02 | |
| ACCEPTANCE CRITERIA:  **Notification system** integrated with the property management platform.  Send notifications for new work orders, maintenance progress, and completion.  **Manage notification settings** to choose types of notifications to receive (email, SMS, or app-based). | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| User Story No: 5 | Tasks: 3 | | Priority: Highest |
| **As a Product Owner**,  I want to ensure that the EWL platform can handle user registration, user roles, and permissions so that only authorized users can access certain features. | | | |
| BV: 600 | | CP: 04 | |
| ACCEPTANCE CRITERIA:  **User registration and role assignment** screen for creating new accounts and assigning roles (e.g., Property Manager, Technician, Tenant, etc.).  Permissions based on roles (e.g., Technicians can only view and update work orders).  **Login page** for user authentication with username and password | | | |

**Document 4: Agile PO Experience**

**Answer:** The **Product Owner (PO)** plays a crucial role in the Agile process, ensuring that the product vision aligns with market needs and business goals. Below is a detailed description of the PO's responsibilities and experiences during the **EWL for Estate Account** project.

### ****Responsibilities of the Product Owner (PO) in a Project****

1. **Market Analysis**
   * **Analysis of Market Need/Demand**: The PO conducts thorough research to understand the market’s needs, identifying what solutions are in demand within the estate management industry.
   * **Availability of Similar Products**: The PO examines existing solutions in the market to identify gaps and opportunities for differentiation.
2. **Enterprise Analysis**
   * **Due Diligence on Market Opportunity**: The PO assesses the market opportunities for the **EWL for Estate Account**, ensuring the project has potential to address real business problems and create value for stakeholders.
3. **Product Vision and Roadmap**
   * **Product Vision**: The PO defines the product vision based on the market and enterprise analysis, ensuring that the product aligns with the needs and strategic direction of the company.
   * **Product Roadmap**: The PO creates a high-level product roadmap outlining key features and their timeline, ensuring that development efforts are aligned with the long-term product goals.
4. **Managing Product Features**
   * **Managing Stakeholder Expectations**: The PO works with stakeholders to ensure their needs are met, while also communicating product limitations and trade-offs.
   * **Prioritization of Epics, Stories, and Features**: The PO prioritizes product features based on their criticality, potential return on investment (ROI), and business needs.
5. **Managing Product Backlog**
   * **Prioritization of User Stories**: The PO organizes the product backlog, ensuring that user stories are prioritized based on their value and importance.
   * **Reprioritization Based on Stakeholder Needs**: The PO adjusts priorities as new feedback or requirements emerge from stakeholders.
   * **Epics Planning**: The PO works with the team to break down large, high-level features (epics) into manageable user stories.
6. **Managing Overall Iteration Progress**
   * **Sprint Progress Review**: The PO regularly reviews the progress of sprints to ensure that development is on track and aligned with product goals.
   * **Reprioritization of Sprints and Epics**: The PO adjusts the sprint backlog and epics as needed to reflect changing priorities.
   * **Sprint Retrospectives with Business Analyst**: The PO participates in sprint retrospectives to evaluate what went well and what could be improved in future iterations.

**Sprint Meetings Managed by the Product Owner**

From this project, the **Product Owner** has gained valuable experience in facilitating and handling various sprint meetings:

1. **Sprint Planning Meeting**: The PO collaborates with the Scrum team to define the goals for the sprint, clarify user stories, and establish the scope of work.
2. **Daily Scrum Meeting**: The PO attends daily scrum meetings to stay informed about the team’s progress, address any obstacles, and provide necessary clarifications.
3. **Sprint Review Meeting**: The PO participates in the sprint review to assess the completed work, gather feedback from stakeholders, and ensure that the product is on track to meet its goals.
4. **Sprint Retrospective Meeting**: The PO takes part in sprint retrospectives to reflect on the sprint process, identify areas for improvement, and implement changes in future sprints.
5. **Backlog Refinement Meeting**: The PO conducts backlog refinement sessions with the Scrum team to ensure that the backlog is well-prioritized and that user stories are ready for upcoming sprints.

### ****User Story Creation and Key Components****

The PO is responsible for creating user stories, which are essential for breaking down product features into actionable tasks. The following elements are included in each user story:

* **Story Number**: Unique identification for each user story.
* **Tasks**: Specific actions or steps needed to implement the user story.
* **Priority**: The level of importance or urgency of the user story (e.g., High, Medium, Low).
* **Acceptance Criteria**: Clear and measurable conditions that determine when the user story is complete and functional.
* **BV (Business Value)**: The estimated value the feature brings to the business (e.g., revenue increase, user satisfaction).
* **CP (Complexity Points)**: A measure of the effort required to implement the user story (e.g., time, resources).

### ****The Product Owner’s Role in Scrum****

In **Scrum**, the **Product Owner** serves as the liaison between various stakeholders and the Scrum team. Their key responsibilities include:

* **Communication with Business Stakeholders**: The PO ensures that business stakeholders’ needs and feedback are communicated to the Scrum team. They help manage expectations and align product goals with business objectives.
* **Collaboration with Scrum Team**: The PO works closely with the Scrum team to ensure they understand the business requirements and priorities. This collaboration helps ensure the development team builds the right product.
* **Defining Product Features**: The PO translates high-level business goals and customer needs into specific, actionable product features, which are then added to the product backlog.
* **Breaking Features into Product Backlog Items**: The PO ensures that each product feature is broken down into smaller, manageable product backlog items (user stories), making it easier for the Scrum team to work on them in iterative sprints.

### ****Lessons Learned from the Project****

Throughout this project, the Product Owner has learned a variety of valuable lessons, including:

* **Handling Sprint Meetings**: The PO has learned the importance of participating in and managing the different sprint meetings. This includes **Sprint Planning**, **Daily Scrums**, **Sprint Review**, **Sprint Retrospectives**, and **Backlog Refinement** to ensure alignment and continuous improvement.
* **Managing Product Backlog**: The PO has gained insights into the importance of maintaining a well-prioritized and regularly refined backlog. Reprioritizing based on stakeholder feedback and business value is critical to delivering the most valuable features first.
* **User Story Creation**: The PO has learned to effectively break down high-level requirements into clear, actionable user stories, including the creation of tasks, setting priorities, defining acceptance criteria, and assigning business value (BV) and complexity points (CP).

#### **Document 5: Product and sprint backlog and product and sprint burndown charts**

**Answer**:

#### **Product Backlog**

#### The **Product Backlog** is a prioritized list of user stories that represent features, tasks, or work to be completed for the product. These items are placed in the backlog by the **Product Owner** and prioritized based on business value (BV), complexity (CP), and urgency.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **User Story ID** | **User Story** | **Tasks** | **Priority** | **BV (Business Value)** | **CP (Complexity Points)** | **Sprint** |
| **US-001** | Property Registration | Create registration form with necessary fields (Property Name, Address, Contact, etc.) | Highest | 500 | 3 | Sprint 1 |
| **US-002** | Task Management Interface | Design UI for task management (create, assign, and update tasks) | High | 450 | 4 | Sprint 1 |
| **US-003** | Maintenance Tracking | Implement tracking system for maintenance requests | Medium | 400 | 2 | Sprint 2 |
| **US-004** | Notification System | Implement property maintenance notifications for tenants and owners | High | 470 | 3 | Sprint 2 |
| **US-005** | User Roles and Permissions | Set up different user roles (Admin, Property Manager, Tenant, etc.) | Highest | 600 | 5 | Sprint 3 |
| **US-006** | Reporting Dashboard | Develop dashboard to view property status and task reports | High | 500 | 4 | Sprint 3 |

#### **2. Sprint Backlog**

The **Sprint Backlog** is a subset of the product backlog, containing user stories and tasks selected for completion during a specific sprint. The **Scrum Team** commits to completing the selected tasks by the end of the sprint.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **User Story ID** | **User Story** | **Tasks** | **Owner** | **Status** | **Estimated Effort (hours)** |
| **US-001** | Property Registration | Design registration form; Implement form fields | Developer 1 | In Progress | 12 |
| **US-002** | Task Management Interface | Design UI; Implement task list view; Add task creation functionality | Developer 2 | Not Started | 16 |
| **US-003** | Maintenance Tracking | Create database schema for tracking; Develop tracking UI | Developer 3 | Not Started | 14 |
| **US-004** | Notification System | Implement email notification system; Integrate with backend | Developer 4 | In Progress | 18 |
| **US-005** | User Roles and Permissions | Define user roles and permissions; Set up user authentication | Developer 5 | Not Started | 20 |

**Product Burndown Chart:**

**Sprint Burndown chart:**