**Project :- Application Tracking System**

**AGILE DOCUMENTS**

**Document 1: Definition of Done**

Creating a project on the **Definition of Done (DoD)** with a detailed checklist is a great way to ensure that your team delivers high-quality work consistently. Below is a structured approach the project.

**Definition :**-

As Per Agile Extension to the BABOK® Guide v2, Definition of Done is a technique where the team agrees on, and prominently displays, a list of criteria which must be met before a backlog item is considered done.

That is the team has to create a well-defined, unambiguous, measurable, agreed-upon, and shared Definition of Done between all team members.

**Project: Definition of Done (DoD) Checklist**

**Objective:**

To establish a clear and shared understanding of what "done" means for any user story, feature, or task in the project. This ensures that all deliverables meet the required quality standards and are ready for release.

**Definition of Done (DoD) Checklist**

The following checklist outlines the criteria that must be met for a feature or user story to be considered "done":

**Produced Code for Presumed Functionalities**

* + Code is written and implements all functionalities as described in the user story or task.
  + Code is clean, readable, and follows the team's coding standards.

**Assumptions of User Story Met**

* + All assumptions and requirements outlined in the user story are fulfilled.
  + Edge cases and potential risks are addressed.

**Project Builds Without Errors**

* + The project compiles and builds successfully in all relevant environments (local, test, staging).
  + No build errors or warnings are present.

1. **Unit Tests Written and Passing**
   * Unit tests are written for all new code and cover critical functionalities.
   * All unit tests pass, and test coverage meets the team's agreed-upon threshold.
2. **Project Deployed on the Test Environment Identical to Production Platform**
   * The feature is deployed to a test environment that mirrors the production environment.
   * Deployment is successful, and the feature is accessible.
3. **Tests on Devices/Browsers Listed in the Project Assumptions Passed**
   * The feature is tested on all devices, browsers, and operating systems specified in the project requirements.
   * No major issues are found during cross-browser or cross-device testing.
   * **Feature** Approved by UX Designer
   * The UX designer reviews the feature and confirms that it meets the design specifications.
   * Any feedback from the UX designer is incorporated.
4. **QA Performed & Issues Resolved**
   * Quality Assurance (QA) testing is completed, and all identified issues are resolved.
   * No critical or high-priority bugs remain open.
5. **Feature Tested Against Acceptance Criteria**
   * The feature is tested against the acceptance criteria defined in the user story.
   * All acceptance criteria are met.
6. **Feature Approved by Product Owner**
   * The Product Owner reviews the feature and confirms that it meets the business requirements.
   * Any feedback from the Product Owner is addressed.
7. **Refactoring Completed**
   * Code is refactored to improve readability, maintainability, and performance.
   * Technical debt is minimized.
8. **Any Configuration or Build Changes Documented**
   * All configuration or build changes are documented and communicated to the team.
   * Documentation is updated in the relevant repositories or knowledge bases.
9. **Documentation Updated**
   * All relevant documentation (e.g., user guides, technical documentation, API docs) is updated to reflect the new feature or changes.
   * Documentation is clear, concise, and accessible to the intended audience.
10. **Peer Code Review Performed**
    * The code has been reviewed by at least one other team member.
    * All feedback from the code review is addressed, and approvals are obtained.
11. **Additional Considerations for the Project**
12. **Team Alignment**
    * Ensure that the entire team (developers, QA, designers, Product Owner, etc.) agrees on the DoD checklist.
    * Conduct a workshop or meeting to discuss and refine the checklist.
13. **Automation**
    * Automate as many steps as possible (e.g., unit tests, build processes, deployment) to reduce manual effort and ensure consistency.
14. **Continuous Improvement**
    * Regularly review and update the DoD checklist based on feedback and lessons learned from previous sprints or projects.
15. **Tooling**
    * Use tools to track and enforce the DoD (e.g., CI/CD pipelines, test automation frameworks, code review tools like GitHub/GitLab).
16. **Communication**
    * Clearly communicate the DoD to all stakeholders to ensure everyone understands the expectations.

**Project Deliverables**

1. **DoD Checklist Document**
   * A finalized version of the DoD checklist that can be shared with the team and stakeholders.
2. **Process Documentation**
   * A guide on how to use the DoD checklist during the development lifecycle.
3. **Training or Workshop**
   * A session to train the team on the DoD checklist and its importance.
4. **Implementation Plan**
   * A plan for integrating the DoD checklist into the team's workflow (e.g., adding it to sprint planning, code review processes, etc.).

**Example Workflow Using the DoD Checklist**

1. **Sprint Planning**
   * Review the DoD checklist with the team to ensure everyone understands the requirements.
2. **Development**
   * Developers write code and ensure it meets the DoD criteria (e.g., unit tests, refactoring).
3. **Code Review**
   * Peer code review is performed, and feedback is addressed.
4. **Testing**
   * QA tests the feature against the acceptance criteria and cross-browser/device requirements.
5. **Deployment**
   * The feature is deployed to the test environment, and final checks are performed.
6. **Approval**
   * The UX designer and Product Owner review and approve the feature.
7. **Documentation**
   * All documentation is updated, and configuration changes are recorded.
8. **Completion**
   * The feature is marked as "done" and ready for release.

**Example DoD for a Specific Feature: "Candidate Resume Upload"**

A user story for **"As a recruiter, I want to upload candidate resumes in PDF format so that I can store them in the ATS"** is considered **Done** when:

The resume upload feature allows only PDF files up to 5MB.  
 Uploaded resumes are stored securely in the database.  
 Recruiters receive confirmation upon successful upload.  
 Validation errors are displayed for unsupported formats or large files.  
 Resume parsing functionality extracts candidate details correctly.  
 Functional, integration, and regression testing are completed.  
 No high-severity bugs remain open.  
 Stakeholders have tested and approved the feature in UAT.

**This DoD ensures the feature is ready for production with minimal risk**.

**Document 2- Product Vision**

**Product Vision for ATS:**"To revolutionize the recruitment process by providing a seamless, AI-driven Applicant Tracking System that empowers organizations to find, engage, and hire top talent efficiently, while delivering an exceptional candidate experience. Our ATS will simplify hiring workflows, reduce bias, and enable data-driven decisions, making it the go-to solution for companies of all sizes to build high-performing teams."

**Key Elements of a Product Vision for Your ATS:**

1. **Purpose:** Why does your ATS exist? (e.g., to streamline hiring, improve candidate experience, or reduce bias in recruitment.)
2. **Target Audience:** Who is it for? (e.g., recruiters, hiring managers, HR teams, or small/medium/large enterprises.)
3. **Core Value:** What unique value does it provide? (e.g., AI-driven insights, automation, integration with other tools, or user-friendly design.)
4. **Long-Term Impact:** What change will it bring to the recruitment industry or its users? (e.g., faster hiring, better candidate matches, or reduced administrative burden.)
5. **Differentiation:** What makes your ATS stand out from competitors? (e.g., advanced analytics, mobile-first design, or affordability for small businesses.)

### Questions to Help Define Your ATS Product Vision:

* What problem are we solving for recruiters and candidates?
* How will our ATS improve the hiring process compared to existing solutions?
* What kind of experience do we want to deliver to recruiters and candidates?
* How will our ATS adapt to future trends in recruitment and technology?
* What is our ultimate goal for this product in 5 or 10 years?
* Here’s how we can structure the product vision for our ATS:

#### 1. Purpose (Why does it exist?)

* Define the core reason for building the ATS. What is the ultimate goal?
* Example: "To transform the recruitment process by making it faster, more efficient, and more inclusive."

#### 2. Target Audience (Who is it for?)

* Identify the primary users and beneficiaries of the ATS.
* Example: "Our ATS is designed for recruiters, hiring managers, HR professionals, and candidates seeking a seamless hiring experience."

#### 3. Problem Statement (What problem does it solve?)

* Highlight the pain points your ATS addresses.
* Example: "Recruiters often struggle with manual resume screening, inefficient communication, and biased hiring processes. Candidates face a lack of transparency and poor communication during the hiring process."

#### 4. Value Proposition (What value does it provide?)

* Explain how your ATS solves the problem and delivers value.
* Example: "Our ATS leverages AI to automate resume screening, reduce bias, and provide real-time updates to candidates, ensuring a smooth and transparent hiring process."

#### 5. Unique Selling Points (How is it different?)

* Highlight what makes your ATS stand out from competitors.
* Example: "Unlike traditional ATS platforms, our solution integrates advanced AI analytics, a mobile-first design, and seamless integration with popular HR tools."

|  |  |  |  |
| --- | --- | --- | --- |
| **Scrum Project Name:** | ATS Implementation Project |  |  |
| **Venue:** | Virtual (Zoom) |  |  |
| **Date : October 25, 2024** | Start time: 9:00 AM | **End time:** 5:00 PM | **Duration:** 8 hours |
| **Client:** | TAP Solution LLP |  |  |
| **Stakeholder list:** | Ravi Singh ( CEO) |  |  |
|  | Smita Pandey ( HR Manager ) |  |  |
|  | Rajesh Singh ( IT Lead ) |  |  |
|  | Mitesh Singh ( BA ) |  |  |
| **Scrum Team** | | | |
| **Scrum Master:** | Rakesh Teja |  |  |
| **Product owner:** | Suman Singh |  |  |
| **Scrum Developer 1:** | Sachin Gutee |  |  |
| **Scrum Developer 2:** | Nikita Meena |  |  |
| **Scrum Developer 3:** | Satyam Singh |  |  |
| **Scrum Developer 4:** | Sai Teja |  |  |
| **Scrum Developer 5:** | Nikita Tiwari |  |  |

**Document 3 User Stories**:

|  |  |  |
| --- | --- | --- |
| User Story 1 :- Resume Parsing and Candidate Profile Creation | Tasks 2 | Priority : Highest |
| As a recruiter, I want to upload resumes and automatically create candidate profiles so that I can save time and reduce manual data entry. | | |
| BV:500 | CP:05 | |
| ACCEPTANCE CRITERIA |  |  |
| 1. The system must accept resumes in PDF, DOCX, and TXT formats. 2. The system must extract key details (e.g., name, contact info, skills, and experience) and populate the candidate profile automatically. 3. Extracted data should be editable in case of errors. 4. The system should flag resumes with missing or unclear information for manual review. 5. A confirmation message should appear after successful profile creation. | | |

|  |  |  |
| --- | --- | --- |
| User Story 2 :- Job Posting Creation | Tasks 2 | Priority : Highest |
| As a recruiter, I want to create and publish job postings so that I can attract qualified candidates. | | |
| BV: 200 | CP:03 | |
| ACCEPTANCE CRITERIA |  |  |
| 1. The system must provide a form with fields for job title, description, location, salary range, and required qualifications. 2. The system should allow hiring managers to save drafts and edit job postings before publishing. 3. Job postings should be publishable to the company’s career page and integrated job boards (e.g., LinkedIn, Indeed). 4. The system should notify the hiring manager when the job posting is live. 5. The system should validate mandatory fields (e.g., job title, description) before publishing. | | |

|  |  |  |
| --- | --- | --- |
| User Story 3 :- ****Interview Scheduling**** | Tasks 2 | Priority : Highest |
| As a recruiter, I want to schedule interviews with candidates so that I can coordinate interviews efficiently. | | |
| BV: 200 | CP:05 | |
| ACCEPTANCE CRITERIA |  |  |
| 1. The system must allow recruiters to select available time slots from their calendar. 2. The system should send an email invitation to the candidate with the interview details (date, time, location/video link). 3. Candidates should be able to confirm, reschedule, or decline the interview via the email invitation. 4. The system should notify the recruiter of the candidate’s response. 5. The system should sync scheduled interviews with the recruiter’s calendar (e.g., Google Calendar, Outlook | | |

|  |  |  |
| --- | --- | --- |
| User Story 4 :- Resume Parsing and Candidate Profile Creation | Tasks 2 | Priority : Highest |
| As a recruiter, I want to upload resumes and automatically create candidate profiles so that I can save time and reduce manual data entry. | | |
| BV:500 | CP:05 | |
| ACCEPTANCE CRITERIA |  |  |
| 1. The system must accept resumes in PDF, DOCX, and TXT formats. 2. The system must extract key details (e.g., name, contact info, skills, experience) and populate the candidate profile automatically. 3. Extracted data should be editable in case of errors. 4. The system should flag resumes with missing or unclear information for manual review. 5. A confirmation message should appear after successful profile creation. | | |

|  |  |  |
| --- | --- | --- |
| User Story 5 :- Job Posting Creation | Tasks 2 | Priority : Highest |
| As a recruiter, I want to create and publish job postings so that I can attract qualified candidates. | | |
| BV: 200 | CP:03 | |
| ACCEPTANCE CRITERIA |  |  |
| 1. The system must provide a form with fields for job title, description, location, salary range, and required qualifications. 2. The system should allow hiring managers to save drafts and edit job postings before publishing. 3. Job postings should be publishable to the company’s career page and integrated job boards (e.g., LinkedIn, Indeed). 4. The system should notify the hiring manager when the job posting is live. 5. The system should validate mandatory fields (e.g., job title, description) before publishing. | | |

|  |  |  |
| --- | --- | --- |
| User Story 6 :- **Interview Scheduling** | Tasks 2 | Priority : Highest |
| As a recruiter, I want to schedule interviews with candidates so that I can coordinate interviews efficiently. | | |
| BV: 200 | CP:05 | |
| ACCEPTANCE CRITERIA |  |  |
| The system must allow recruiters to select available time slots from their calendar.  The system should send an email invitation to the candidate with the interview details (date, time, location/video link).  Candidates should be able to confirm, reschedule, or decline the interview via the email invitation.  The system should notify the recruiter of the candidate’s response.  The system should sync scheduled interviews with the recruiter’s calendar (e.g., Google Calendar, Outlook | | |

|  |  |  |
| --- | --- | --- |
| User Story 7 :- Reporting and Analytics | Tasks 2 | Priority : Highest |
| As an HR manager, I want to generate reports on hiring metrics so that I can track recruitment performance | | |
| BV: 500 | CP:08 | |
| ACCEPTANCE CRITERIA |  |  |
| 1. The system must provide pre-built reports for metrics like time-to-hire, cost-per-hire, and candidate pipeline. 2. The system should allow HR managers to customize reports by selecting specific date ranges, job roles, or recruiters. 3. Reports should be exportable in PDF and Excel formats. 4. The system should display data visually using charts and graphs (e.g., bar charts, pie charts). 5. The system should update reports in real-time as new data is added. | | |

|  |  |  |
| --- | --- | --- |
| User Story 8 :- Resume Parsing | Tasks 2 | Priority : Highest |
| As a recruiter, I want resumes to be automatically parsed so that I can quickly extract candidate details. | | |
| BV: 200 | CP:11 | |
| ACCEPTANCE CRITERIA |  |  |
| System should extract name, contact details, experience, skills, and education from resumes.  Parsed data should be auto-mapped to the candidate profile fields.  Recruiters should have the ability to manually edit parsed details.  System should support PDF and DOCX formats for resume uploads.  Parsing accuracy should be at least **85%** for structured resumes. System should notify recruiters if parsing fails or is incomplete.  Processing time for parsing should not exceed **3 seconds per resume**.  GDPR compliance must be ensured when storing parsed data. | | |

|  |  |  |
| --- | --- | --- |
| User Story 9:- Candidate Communication (Email Integration) | Tasks 2 | Priority : Medium |
| As a recruiter, I want to send emails to candidates directly from the ATS so that I can streamline communication | | |
| BV: 100 | CP:02 | |
| ACCEPTANCE CRITERIA |  |  |
| 1. The system must allow recruiters to compose and send emails to candidates from within the ATS. 2. The system should provide email templates for common scenarios (e.g., interview invitations, rejections). 3. Recruiters should be able to track whether an email has been opened by the candidate. 4. Sent emails should be saved in the candidate’s profile for future reference. 5. The system should support bulk emailing for multiple candidates. 6. The system should update reports in real-time as new data is added. | | |

|  |  |  |
| --- | --- | --- |
| User Story 10 :- Candidate Status Tracking | Tasks 2 | Priority : Highest |
| As a recruiter, I want to track the status of candidates in the hiring pipeline so that I can manage the recruitment process effectively | | |
| BV: 200 | CP:08 | |
| ACCEPTANCE CRITERIA |  |  |
| 1. The system must allow recruiters to update candidate statuses (e.g., Applied, Interviewing, Offered, Hired, Rejected). 2. The system should display the current status of each candidate in the pipeline. 3. Recruiters should be able to filter candidates by status. 4. The system should notify recruiters when a candidate’s status has been unchanged for a specified period. 5. The system should provide a visual representation of the hiring pipeline | | |

|  |  |  |
| --- | --- | --- |
| User Story 11 Collaboration Tools for Hiring Teams | Tasks 2 | Priority : Low |
| As a hiring team member, I want to leave comments and feedback on candidate profiles so that I can collaborate with my team effectively. | | |
| BV: 200 | CP:4 | |
| ACCEPTANCE CRITERIA |  |  |
| 1. The system must allow team members to add comments to candidate profiles. 2. Comments should be visible to all team members with access to the candidate’s profile. 3. The system should support @mentions to notify specific team members. 4. Team members should be able to edit or delete their own comments. 5. The system should display a timestamp for each comment. | | |

|  |  |  |
| --- | --- | --- |
| User Story 12 Integration with HRIS Systems | Tasks 2 | Priority : Medium |
| As an HR manager, I want to integrate the ATS with our HRIS system so that I can sync candidate data after hiring. | | |
| BV: 200 | CP:10 | |
| ACCEPTANCE CRITERIA |  |  |
| The system must support integration with popular HRIS systems (e.g., Workday, SAP SuccessFactors).The system should automatically transfer candidate data (e.g., name, contact info, job role) to the HRIS upon hiring.The system should log successful and failed sync attempts for troubleshooting.The system should allow HR managers to manually trigger a sync if needed.The system should validate data before syncing to avoid errors. | | |

|  |  |  |
| --- | --- | --- |
| User Story 13 **Automated Candidate Screening** | Tasks 2 | Priority : Highest |
| As a recruiter, I want the system to automatically screen candidates based on predefined criteria so that I can focus on the most qualified applicants.. | | |
| BV: 200 | CP:08 | |
| ACCEPTANCE CRITERIA |  |  |
| The system must allow recruiters to define screening criteria (e.g., skills, experience, education).  The system should automatically rank candidates based on how well they match the criteria.  The system should flag candidates who meet a minimum threshold for further review.  Recruiters should be able to override the system’s ranking if needed.  The system should provide a report explaining why a candidate was ranked a certain way. | | |

|  |  |  |
| --- | --- | --- |
| User Story 14 Job Posting | Tasks 2 | Priority : Highest |
| As a Recruiter, I want to create and publish job postings with relevant details, So that I can attract and collect applications from suitable candidates. | | |
| BV: 200 | CP:05 | |
| ACCEPTANCE CRITERIA |  |  |
| Recruiters can enter job title, description, requirements, and location. ✅ Recruiters can set the application deadline. ✅ The job post should be visible to applicants once published | | |
| User Story Candidate Profile Submission | Tasks 2 | Priority : Medium |
| **As a** Job Applicant, **I want to** submit my application along with my resume and cover letter, **So that** I can apply for job openings easily. | | |
| BV: 200 | CP:06 | |
| ACCEPTANCE CRITERIA |  |  |
| Candidates can upload resumes in multiple formats (PDF, DOCX). ✅ The system validates mandatory fields before submission. ✅ Candidates receive an email confirmation upon successful application. | | |

|  |  |  |
| --- | --- | --- |
| User Story 15 ****Onboarding Workflow Initiation**** | Tasks 2 | Priority : Medium |
| As a HR administrator,  I want the ATS to trigger onboarding workflows once a candidate accepts an offer,  So that the transition from hiring to onboarding is seamless. | | |
| BV: 200 | CP:04 | |
| ACCEPTANCE CRITERIA |  |  |
| Candidates can upload resumes in multiple formats (PDF, DOCX). ✅ The system validates mandatory fields before submission. ✅ Candidates receive an email confirmation upon successful application. | | |
| User Story ****Candidate Status Tracking**** | Tasks 2 | Priority : Medium |
| As a recruiter,  I want to update and track the status of candidates (e.g., Applied, Interviewing, Offered, Hired, Rejected),  So that I can manage the hiring pipeline effectively | | |
| BV: 100 | CP:02 | |
| ACCEPTANCE CRITERIA |  |  |
| * AC1: Status options are customizable by the admin. * AC2: Recruiters can update the status manually or automatically (e.g., after an interview). * AC3: The system logs status changes with timestamps and user details. | | |

|  |  |  |
| --- | --- | --- |
| User Story 16 ****Customizable Hiring Workflows**** | Tasks 2 | Priority : Medium |
| * **As a** hiring manager, * **I want** to create and customize hiring workflows for different job roles, * **So that** I can ensure the hiring process aligns with the role’s requirements. | | |
| BV: 100 | CP:02 | |
| ACCEPTANCE CRITERIA |  |  |
| * Workflows can include stages like Screening, Interview, Assessment, Offer, etc. * AC2: Admins can add, remove, or reorder stages. * AC3: Notifications are sent to relevant stakeholders at each stage   . | | |

|  |  |  |
| --- | --- | --- |
| User Story 17 ****Candidate Duplicate Detection**** | Tasks 2 | Priority : Medium |
| * **As a** recruiter, * **I want** the system to detect and flag duplicate candidate profiles, * **So that** I can avoid confusion and maintain clean data. | | |
| BV: 100 | CP:03 | |
| ACCEPTANCE CRITERIA |  |  |
| * AC1: The system checks for duplicates based on email, phone number, and name. * AC2: Recruiters can merge duplicate profiles manually. * AC3: A log of merged profiles is maintained for audit purposes. * . | | |

|  |  |  |
| --- | --- | --- |
| User Story 18 ****Diversity and Inclusion Reporting**** | Tasks 2 | Priority : Medium |
| * **As a** HR manager, * **I want** to generate reports on diversity metrics (e.g., gender, ethnicity) for candidates and hires, * **So that** I can ensure our hiring practices align with organizational goals.   . | | |
| BV: 100 | CP:03 | |
| ACCEPTANCE CRITERIA |  |  |
| * AC1: Candidates can voluntarily provide diversity data during the application process. * AC2: Reports can be filtered by job role, department, and time period. * AC3: Data is anonymized for privacy compliance. | | |

|  |  |  |
| --- | --- | --- |
| User Story 19 ****Mobile-Friendly Application Process**** | Tasks 2 | Priority : Medium |
| * **As a** candidate, * **I want** to apply for jobs using a mobile-friendly interface, * **So that** I can easily apply from my smartphone or tablet.   . | | |
| BV: 100 | CP:04 | |
| ACCEPTANCE CRITERIA |  |  |
| * AC1: The application form is responsive and works on all devices. * AC2: Candidates can upload documents and complete assessments on mobile. * AC3: The process is optimized for speed and usability. * . | | |

|  |  |  |
| --- | --- | --- |
| User Story 20 ****Compliance and Audit Logs**** | Tasks 2 | Priority : Medium |
| * **As a** HR administrator, * **I want** the system to maintain detailed logs of all actions taken in the ATS, * **So that** I can ensure compliance with legal and organizational policies.   . | | |
| BV: 100 | CP:03 | |
| ACCEPTANCE CRITERIA |  |  |
| * AC1: Logs include user actions, timestamps, and changes made. * AC2: Admins can export logs for audit purposes. * AC3: Sensitive actions (e.g., deleting a candidate profile) require additional approval. | | |

|  |  |  |
| --- | --- | --- |
| User Story 21 ****Talent Pool Management**** | Tasks 2 | Priority : Medium |
| * **As a** recruiter, * **I want** to add candidates to a talent pool for future opportunities, * **So that** I can maintain a pipeline of qualified candidates.   . | | |
| BV: 100 | CP:03 | |
| ACCEPTANCE CRITERIA |  |  |
| * AC1: Recruiters can manually add candidates to the talent pool. * AC2: Candidates are automatically added based on predefined criteria (e.g., skills, experience). * AC3: Recruiters can search and filter the talent pool for specific roles. | | |

### ****Document 4: Agile PO Experience****

### ****Responsibilities of a Product Owner in Agile****

### ****1. Introduction****

* Briefly introduce the role of a Product Owner in Agile methodologies (e.g., Scrum, Kanban).
* Explain why the PO is crucial for bridging the gap between stakeholders, the development team, and the end-users.
* Highlight the importance of the PO's responsibilities in delivering value to the business and customers.

### ****2. Key Responsibilities of a Product Owner****

#### ****2.1 Market Analysis****

* **Objective:** Understand the market, competitors, and customer needs.
* **Activities:**
  + Conduct market research to identify trends and opportunities.
  + Analyze competitors' products and strategies.
  + Gather customer feedback and insights.
* **Outcome:** Use market analysis to inform product decisions and ensure the product meets market demands.

#### ****2.2 Enterprise Analysis****

* **Objective:** Align the product with the organization's goals and strategy.
* **Activities:**
  + Understand the organization's vision, mission, and objectives.
  + Identify business needs and constraints.
  + Ensure the product delivers value to the organization.
* **Outcome:** A product that supports the organization's strategic goals.

#### ****2.3 Product Vision and Roadmap****

* **Objective:** Define the long-term direction of the product.
* **Activities:**
  + Create a clear and inspiring product vision.
  + Develop a product roadmap that outlines key milestones and deliverables.
  + Communicate the vision and roadmap to stakeholders and the team.
* **Outcome:** A shared understanding of the product's direction and priorities.

#### ****2.4 Managing Product Features****

* **Objective**: Prioritize and manage the features that deliver the most value.
* **Activities**:
  + Collaborate with stakeholders to identify and define features.
  + Prioritize features based on business value, customer needs, and technical feasibility.
  + Ensure features are well-defined with clear acceptance criteria.
* **Outcome**: A product backlog with high-value features ready for development.

#### ****2.5 Managing Product Backlog****

* **Objective**: Maintain a well-organized and prioritized product backlog.
* **Activities**:
  + Continuously refine and prioritize backlog items.
  + Break down epics into user stories and tasks.
  + Ensure the backlog is transparent and understood by the team.
* **Outcome**: A backlog that guides the team in delivering value incrementally.

#### ****2.6 Managing Overall Iteration Progress****

* **Objective**: Ensure the team is on track to meet iteration goals.
* **Activities**:
  + Monitor progress during sprints.
  + Address any blockers or issues that arise.
  + Collaborate with the Scrum Master and team to ensure smooth execution.
  + **Outcome**: Successful delivery of iteration goals.

#### ****2.7 Sprint Progress Review****

* **Objective**: Evaluate the progress made during the sprint.
* **Activities**:
  + Participate in sprint reviews to demonstrate completed work.
  + Gather feedback from stakeholders.
  + Adjust plans based on feedback and progress.
* **Outcome**: Stakeholder alignment and continuous improvement.

#### ****2.8 Reprioritization of Sprints and Epics (if needed)****

* **Objective**: Adapt to changing priorities and requirements.
* **Activities**:
  + Reassess priorities based on new information or feedback.
  + Adjust the sprint backlog and roadmap as needed.
  + Communicate changes to the team and stakeholders.
* **Outcome:** A product that remains aligned with business needs and market conditions.

#### ****2.9 Sprint Retrospectives with Business Analyst****

* **Objective**: Reflect on the sprint and identify areas for improvement.
* **Activities**:
  + Collaborate with the Business Analyst and team to discuss what went well and what didn’t.
  + Identify actionable improvements for the next sprint.
  + Ensure continuous learning and growth within the team.
* **Outcome**: A more efficient and effective team.
* **From this project I have learned how to handle sprint meetings such as**

## **Business Analyst Role in Sprint Meetings – ATS Project**

### ****1. Sprint Planning Meeting****

* Reviewed the product backlog with the Product Owner to ensure user stories were well-defined and prioritized.
* Clarified requirements, dependencies, and acceptance criteria for the development team.
* Collaborated with developers and testers to break down complex stories into smaller tasks.
* Ensured Business Value (BV) and Complexity Points (CP) were assigned correctly for better sprint planning.

**Example:**  
"*For the ATS project, I worked with the Product Owner to refine the 'Candidate Profile Creation' user story. I ensured mandatory fields, validation rules, and integration with resume parsing were clearly defined before Sprint Planning, reducing last-minute changes."*

### ****2. Daily Scrum Meeting****

* Attended stand-up meetings to track progress and identify requirement-related blockers.
* Provided quick clarifications on business rules or system behavior when needed.
* Collaborated with developers and testers to resolve requirement gaps on time.
* Escalated dependencies or issues to stakeholders to prevent delays.

**Example:**  
*"During a sprint, developers were unclear about how the ATS should handle duplicate candidate profiles. I quickly coordinated with stakeholders, gathered clarification, and updated the user story, ensuring minimal disruption to development."*

### ****3. Sprint Review Meeting****

* Assisted in demonstrating completed features to stakeholders and collected feedback.
* Verified if the implemented functionality met business requirements and user expectations.
* Documented stakeholder feedback and discussed necessary refinements with the Product Owner.

**Example:**  
"In one Sprint Review, stakeholders wanted an additional filter for job applications in the ATS dashboard. I documented this feedback and added it to the backlog for prioritization in the next sprint."

### ****4. Sprint Retrospective Meeting****

* Analyzed what worked well and identified areas of improvement in the requirement gathering process.
* Suggested improvements, such as conducting stakeholder pre-demo sessions to minimize rework.
* Contributed to enhancing cross-functional collaboration by addressing communication gaps.

**Example:**  
"A recurring issue in retrospectives was late feedback from stakeholders. I proposed conducting informal stakeholder previews before Sprint Reviews, which significantly reduced last-minute change requests."

### ****5. Backlog Refinement Meeting****

* Continuously refined and prioritized user stories based on business goals and stakeholder feedback.
* Ensured each user story had a clear description, acceptance criteria, and dependencies addressed.
* Worked with developers to break down high-complexity stories and improve sprint readiness.

**Example:**  
"A job posting automation feature in the ATS had multiple dependencies. During backlog refinement, I collaborated with developers to split the story into two: one for core functionality and another for API integration, ensuring smoother sprint execution."

### In Scrum, a product owner serves as the liaison between multiple areas of an organization. This person communicates with business stakeholders and collaborates closely with Scrum teams to keep all areas of the business informed on a project's development.

### The product owner develops a vision of a product's function and operation, which in turn allows this Scrum team member to define product features and break those features into product backlog items.

* **Also, User stories creation and what things will be included in user stories such as**

## **Business Analyst Role in User Story Creation – ATS Project**

### ****User Story Creation Process:****

As a Business Analyst, I played a crucial role in defining and structuring user stories to ensure clarity and alignment with business goals. Each user story included:

* **Story No:** Unique identifier for tracking (e.g., ATS-101).
* **Tasks:** Breakdown of development and testing activities.
* **Priority:** Defined using MoSCoW (Must-have, Should-have, Could-have, Won’t-have).
* **Acceptance Criteria:** Clear conditions to consider the story "done."
* **Business Value (BV):** Impact on business goals (scale of 1-10).
* **Complexity Points (CP):** Effort estimation using story points (e.g., Fibonacci sequence).

### ****Example User Story for ATS Project****

**User Story:**  
"As a recruiter, I want to create and update candidate profiles so that I can track and manage applications efficiently."

* **Story No:** ATS-101
* **Tasks:**
  + Design UI for profile creation
  + Implement backend for data storage
  + Validate required fields
  + Integrate resume parsing feature
* **Priority:** Must-have
* **Acceptance Criteria:**
  + Recruiters should be able to add, edit, and delete candidate details.
  + All required fields (Name, Email, Phone, Resume) must be validated.
  + A confirmation message should appear upon successful profile creation.
* **Business Value (BV):** 8 (High impact on recruitment efficiency)
* **Complexity Points (CP):** 5 (Moderate complexity)

### ****Challenges & Solutions:****

✅ **Challenge:** Requirements were sometimes unclear, leading to confusion among developers.  
✔ **Solution:** I conducted backlog refinement sessions to ensure all user stories were clear and complete.

✅ **Challenge:** Changing priorities from stakeholders impacted sprint planning.  
✔ **Solution:** I used the MoSCoW technique to align priorities with business goals.

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

* + 1. **Product Backlog**

**Definition:-** The Product Backlog is a prioritized list of all features, enhancements, bug fixes, and technical tasks required for the product. It is managed by the Product Owner and continuously refined based on business priorities and stakeholder feedback.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| User story ID | User story | Tasks | Priority | BV | CP | Sprint |
| US-001 | As a user, I want to log in so that I can access my account. | 1. Design login page. | High | 8 | 5 | Sprint 1 |
| 2. Implement authentication logic. |
| 3. Test login functionality. |
| US-002 | As a user, I want to reset my password so that I can regain access to my account. | 1. Design password reset page. | Medium | 6 | 8 | Sprint 2 |
| 2. Implement email verification. |
| 3. Test password reset flow. |
| US-003 | As a user, I want to view my profile so that I can see my personal information. | 1. Design profile page. | Low | 5 | 3 | Sprint 3 |
| 2. Fetch and display user data. |
| 3. Test profile functionality. |
| US-004 | As a user, I want to search for products so that I can find what I need quickly. | 1. Design search bar. | High | 9 | 7 | Sprint 1 |
| 2. Implement search logic. |
| 3. Test search functionality. |
| US-005 | As a user, I want to add items to my cart so that I can purchase them later. | 1. Design cart icon. | High | 10 | 6 | Sprint 2 |
| 2. Implement add-to-cart functionality. |
| 3. Test cart functionality. |

* + 1. **Sprint Backlog**

**Definition :-** The Sprint Backlog is a subset of the Product Backlog that includes the user stories, tasks, and bug fixes selected for a specific sprint. It is a short-term, actionable plan that guides the development team during the sprint

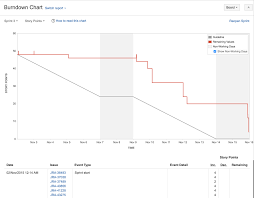
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **User Story ID** | **User Story** | **Tasks** | **Owner** | **Status** | **Estimated Effort (hours)** |
| US-001 | As a user, I want to log in so that I can access my account. | 1. Design login page. | John | In Progress | 8 |
| 2. Implement authentication logic. |
| 3. Test login functionality. |
| US-004 | As a user, I want to search for products so that I can find what I need quickly. | 1. Design search bar. | Sarah | To Do | 12 |
| 2. Implement search logic. |
| 3. Test search functionality. |
| US-002 | As a user, I want to reset my password so that I can regain access to my account. | 1. Design password reset page. | Mike | Done | 10 |
| 2. Implement email verification. |
| 3. Test password reset flow. |

### Tips for Maintaining a Sprint Backlog

* **Keep it Visible**: Use tools like Jira, Trello, or a physical board to make the Sprint Backlog accessible to the team.
* **Update Regularly**: Review and update the backlog during daily stand-ups to reflect the current status.
* **Focus on Collaboration:** Encourage team members to communicate and help each other to ensure tasks are completed on time.
* **Monitor Progress**: Use burndown charts or other metrics to track progress and identify potential bottlenecks.
  + 1. **Product Burndown**

## **What is a Product Burndown Chart?**

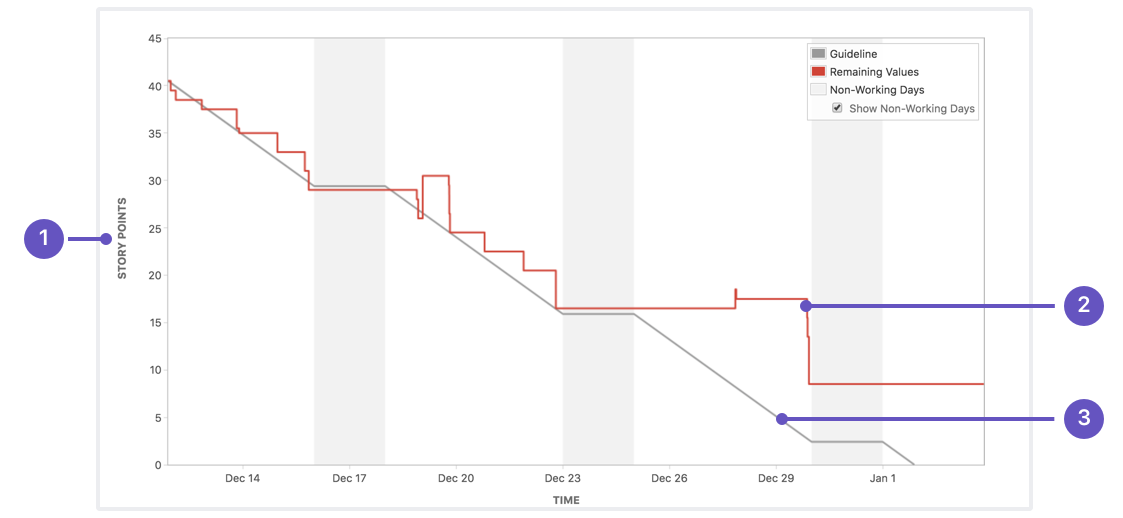
* **Purpose**: To track the completion of work (e.g., user stories, tasks) over time and predict whether the team will meet the sprint or project goals.
* **X-Axis**: Time (e.g., days of the sprint or project timeline).
* **Y-Axis**: Remaining work (e.g., story points, tasks, or hours).
* **Lines**:
  + **Ideal Burndown**: A straight line representing the ideal rate of work completion.
  + **Actual Burndown**: A line showing the actual progress of the team.

****

* + 1. **Sprint Burndown Chart**

## **What is a Sprint Burndown Chart?**

* **Purpose**: To track the completion of work (e.g., tasks, story points) during a sprint and predict whether the team will meet the sprint goal.
* **X-Axis**: Time (e.g., days of the sprint).
* **Y-Axis**: Remaining work (e.g., story points, tasks, or hours).
* **Lines**:
  + **Ideal Burndown**: A straight line representing the ideal rate of work completion.
  + **Actual Burndown**: A line showing the actual progress of the team.

****

**Document 6: Sprint meetings**

**Meeting Type 1: Sprint Planning meeting**

|  |  |
| --- | --- |
| **Date** | October 30, 2024 |
| **Time** | 10:00 AM - 11:30 AM |
| **Location** | Zoom (Zoom Link]) |
| **Prepared By** | Mitesh Singh (BA) |
| **Attendees** | * Smita Singh (Product Owner) * Joginder Sharma (Scrum Master) * Rakesh (Developer) * Kalyani Tatte (QA Engineer) * Sachin Gutte(UX Designer) |

### Agenda:

1. **Review of Previous Sprint**
   * Key accomplishments
   * Unfinished tasks (if any)
   * Lessons learned
2. **Sprint Goals**
   * Define the objectives for the upcoming sprint
3. **Backlog Review**
   * Prioritize user stories/tasks
   * Clarify requirements and acceptance criteria
4. **Capacity Planning**
   * Team availability (vacations, holidays, etc.)
   * Estimate effort for each task
5. **Task Assignment**
   * Assign tasks to team members
6. **Risks and Dependencies**
   * Identify potential risks and dependencies
7. **Q&A and Clarifications**
   * Address any questions or concerns
8. **Next Steps**
   * Confirm sprint start and end dates
   * Schedule daily stand-ups and other meetings

**Agenda Topics**

|  |  |  |
| --- | --- | --- |
| 1. Review of Previous Sprint | Smita Singh | 10 minutes |
| 2. Sprint Goals | Joginder Sharma | 15 minutes |
| 3. Backlog Review | Sachin Gutee | 20 minutes |
| 4. Capacity Planning | Kalynai | 10 minutes |
| 5. Task Assignment | Sushimita | 15 minutes |
| 7. Q&A and Clarifications | Rajeshwari Singh | 10 minutes |

**Other Information**

|  |  |
| --- | --- |
| **Observers** | * + Pratibha ( Stakeholder)   + Sushmita (Intern) |
| **Resources** | * + Jira Board: [Insert Jira Link]   + API Documentation: [Insert API Doc Link]   + Design Mockups: [Insert Figma Link] |
| **Special Notes** | * + Ensure all team members review the updated API documentation before the sprint begins.   + Joginder to hand over QA responsibilities temporarily to Sarah during his leave.   + Mid-sprint review scheduled for November 6, 2023, at 11:00 AM. |

**Meeting Type 2: Sprint review meeting**

|  |  |
| --- | --- |
| **Date** | November 13, 2024 |
| **Time** | 2:00 PM - 3:00 PM |
| **Location**  **Prepared By** | **Zoom ( Link )**  **Mitesh Singh (BA)** |
| **Attendees** | Smita Singh |
| Joginder Sharma |
| Sachin Gutee |
| Kalynai |
| Sushimita |
| Rajeshwari Singh |

|  |  |  |  |
| --- | --- | --- | --- |
| **Sprint Status** | **Things to do** | **Quick Updates** | **What’s Next** |
| **Sprint Status**  **Completed:** API integration, new checkout flow development, and initial QA testing.  **In Progress:** User profile redesign (80% complete).  **Blockers:** Minor bugs in the checkout flow need to be resolved before deployment. | **Sachin:** Demo of the completed API integration.  **Kalyani:** Showcase the new checkout flow design.  **Nikita:** Present QA results and testing outcomes. | API documentation delays were resolved, but impacted the early sprint timeline.  QA resources were limited due to Alex’s temporary leave, but the team adapted well. | Resolve minor bugs in the checkout flow before deployment.  Complete the user profile redesign in the next sprint.  Schedule a retrospective meeting to discuss improvements for future sprints |

**Meeting Type 3: Sprint retrospective meeting**

|  |  |
| --- | --- |
| **Date** | November 13, 2024 |
| **Time** | 2:00 PM - 3:00 PM |
| **Location**  **Prepared By** | **Zoom ( Link )**  **Mitesh Singh (BA)** |
| **Attendees** | Smita Singh |
| Joginder Sharma |
| Sachin Gutee |
| Kalyani |
| Sushimita |
| Rajeshwari Singh |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Agenda** | **What Went Well** | **What Didn’t Go Well** | **Questions** | **Reference** |
| Discuss team progress | Example: Good collaboration, met sprint goal | Example: Scope creep, unclear requirements | How can we improve requirement clarity? | Scrum Guide, Sprint Retrospective Best Practices |
| Review challenges | Efficient backlog grooming | Delays in testing | How can we ensure timely testing? | Agile Manifesto |
| Identify improvements | Effective stakeholder communication | Lack of resources | How can we optimize resource allocation? | Previous retrospective reports |
| Action items planning | Clear sprint planning | Unclear roles in a task | How can we define roles better? | RACI Matrix |

**Meeting Type 4: Daily Stand-up meeting**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Question** | **Name /Role** | **Week "12" (from 18-03-2025 to 24-03-2025)** | | | | |
| **Monday** | **Tuesday** | **Wednesday** | **Thursday** | **Friday** |
| **What did you do yesterday?** | Developer 1 | Worked on backend API integration. | Completed API testing and documentation review. | Integrated API with frontend. | Completed unit testing. | Deployed API to staging and tested endpoints. |
| Developer 2 | Fixed UI bugs in the dashboard. | Implemented new UI components. | Fixed responsive issues. | Finalized UI updates. | Reviewed UI changes with stakeholders. |
| Developer 3 | Updated test cases for new features. | Finished regression testing and reported issues. | Helped with bug fixing. | Submitted test reports. | Conducted performance testing. |
| **What will you do today?** | Developer 1 | Continue API development and testing | Start integration with frontend. | Conduct unit testing for API endpoints | Deploy API to staging | Address feedback from API testing. |
| Developer 2 | Implement new UI enhancements. | Work on responsive design fixes. | Finalize UI refinements. | Conduct UI review with stakeholders | Make final UI adjustments. |
| Developer 3 | Execute regression testing. | Assist with bug fixes. | Prepare test reports. | Conduct UI review with stakeholders | Validate performance testing results. |
| **What (if any) is blocking your progress?** | Developer 1 | Waiting for API documentation update. | No blockers. | No blockers | No blockers. | No blockers. |
| Developer 2 | Need confirmation on new UI designs. | Awaiting feedback from UX team. | Awaiting stakeholder approval. | Waiting for stakeholder feedback. | Awaiting final stakeholder approval |
| Developer 3 | No blockers. | No blockers. | No blockers. | No blockers. | No blockers. |