Key Dates

00 - RE WIN Release Date Calculation.xlsx

Key activities (inc. meetings, events, & actions)

Quarterly planning & prioritisation

- High level planning & prioritising of the quarter between PO and Business Sponsor, with approximate sprints for stories already determined to fall in that quarter

Sprint prioritisation

- Done by PO throughout, but particularly for the next sprint ahead of user story review
- Key activities:
    - Move stories tagged Carry Over into the next sprint
    - Drag and drop in the sprint view to prioritise stories
    - Ensure Priority field is correct for stories
- Generally, items above the 'Bug Fixes' story are ones we intend to complete. Items below are ones we will complete if time.

User story review prep/ sprint planning

- PO (and business sponsor, if req'd):
    - Review stories for the next sprint and confirm priorities
    - Move items out to later sprint(s) if required based on assessment of capacity and initial thoughts on effort for unestimated stories
        - Aim is for a total story point value of 200-300 (depending on number of development days), on the assumption c.100 will be delivered and the remainder rolled over
    - Ensure stories contain the below, and indicate using Ready for Analysis tag
        - An accurate description of what we're trying to do and why
        - Who will be determining the Solution
        - Anyone to involve in UAT, other than the team member the UAT task is assigned to
        - Who to contact for questions and approval, if required (distinguish between questions and approval)
        - Any stakeholders requiring a demo, excluding the team, and whether they are required or optional attendees
        - Any applicable notes, e.g. for background or potential solutions already considered
    - Assign stories
    - Tag stories (key tags: Change, Release Notes, Training, Communication, UAT, Demo, Data, Ready for Analysis, Ready for Build, RN Written)
    - Create and assign tasks, or request team leaders assign tasks
- Team:

    - Add story points for stories assigned to you
    - Review tags to ensure key tags related to role are present
    - Create tasks if not already created
    - Add detail and estimations (original & time remaining) for tasks assigned to you
    - Raise any questions in the discussion section of stories

User Story Review & Planning

- Resolve any missing estimations on tasks or stories
- Team member that a story is assigned to talks through the story, its purpose, any links or dependencies
- Team raise any questions as required and answer these or arrange offline follow-ups to answer
- For stories tagged Carry Over, re-review the story point value for stories and the hours estimates for tasks, and update if required
- Confirm tags are correct

Huddles

- Refer to the dashboard for key queries
- Typical order:
    - Review items closed off since we last met – any key call-outs or dependencies
    - Review any items that are not being actively worked on as not yet started or on hold, and whether we are ready to (re) activate these
    - Review any items future dated and whether any are due or we have received an update earlier than expected
    - Review items currently being worked on, identify any blockers, highlight progress, and update tags etc.
- See the 'Dashboard' section for dev cut-off dates and which sections to use when.
    - Before dev-cut off, use the 'All' section
    - After dev cut-off, start with the non-carry over then move on to carry over if time
        - See tags for Carry Over definition

Demo plan & demo

- Ahead of demo planning:
    - Create a new folder in the sprint demos folder for the appropriate sprint, if required
    - Open the demo planning deck from last sprint, save as new and update title and release date
    - Update the demo content slide it includes all items tagged Demo that are not tagged Carry Over
    - Solution demos should be completed throughout the month on key items, so by the time demo planning occurs key solution demos are complete and the direction of dev has been confirmed
    - Note: for WalkMe, Hannah and Rutuja complete the solution review separately

    - Identify for each item to be demo'd who the audience is – stakeholders or team. The team should join or review the stakeholder demo, where this takes place, to avoid duplication of content.
- During the demo planning call:
    - For each item, identify who will demo and how long this will require
    - Grey and strike out any items that will not be ready to demo and are expected to form part of next month's demo
- After the demo planning call:
    - Send a stakeholder demo invitation if required
        - Invitees are based on the content to be demo'd.
        - Note continuous demos should be conducted with stakeholders throughout for major/ impactful features. The stakeholder demo is showing off a product, not gathering feedback.
        - Feedback received during a wider stakeholder demo should be assessed as either an enhancement or sufficient to hold release – the latter should be a rare occurrence as feedback will have been gathered and acted on throughout development.
    - Share link to demo plan deck with team
    - Update demo call length, if required

Runbook planning

- Ahead of the runbook planning call:
    - Create a new folder in the Release Documentation folder for the appropriate sprint, if required
    - Open the runbook from last sprint and update the content on the cover sheet to reflect the correct completion date and times in UK hours (CST will then update accordingly)
    - Update the link to the Sharepoint folder for that release
    - Review the runbook tab and manually overwrite the Start Date/Time (UK Hours) if the default is incorrect. Otherwise, set the Start Date/ Time (UK Hours) to the default.
    - Mark status of Excluded for any rows that are not applicable from (e.g. if there ae no WalkMe or Power BI activities this month)
    - Set the Status of each item to Not Started
    - Add any comments already aware of relating to e.g. data loads (what data needs to be loaded?) and further info on what the activity encompasses (e.g. 'verify Advocate values match before and after the data load in Power BI reports')
- During the runbook planning call:
    - Work through pre-requisites, pre-deployment activities, deployment activities, and post-deployment activities to ensure the correct items are captured
    - This should include a list of key activities and handover points required at a high level. Post-deployment activity details are captured separately.
        - Activities may change dependent on inclusion of Power BI changes, WalkMe changes, addition of training content, etc. These are generally post-deployment activities and earlier items remain consistent.

        - For info on sample items for inclusion when a release has Power BI or WalkMe impact, refer to earlier runbooks where this activity has been captured.
        - Ensure named resources from Power BI team and DBAs, if required, are available on the targeted date(s).
        - Ensure named resources to update WalkMe content, if required, are available on the targeted date(s).
        - Note for any updates to training content in the form of PDFs (QRGs), this needs to be done in WIN via WalkMe AND in the resource SharePoint
    - Agree start times, durations, and named resources for each key activity
- After runbook planning call:
    - Share link with team

Release Notes

- PO reviews release notes on the day before release
- Confirm all items are included and correct
- Release notes are sent from the team mailbox after release success is confirmed by the colleague identified on the runbook
- If there are any additional comms required as part of the release, these follow the same process, with additional stakeholder sign off before being passed to internal comms or sent from the team mailbox as required. These should have a separate action on the runbook

Release

- Named resources for each runbook activity should update the runbook task as work progresses – Not Started, In Progress, or Complete
- Ensure all Runbook items are updated if the expected time has passed. If not Completed, identify follow up actions and log in DevOps as appropriate.
- QA should tag key individuals in any bugs identified during PPV
- Business admin should tag key individuals in any bugs identified during PDV
- Bugs should be reviewed as they are raised and determined if they need resolution for the release to continue

Post-release

- PO:
    - Removes Carry over tag from what is now the current sprint
    - Once PPV and PDV has been completed, and any additional verification of required checks are successful, update AJGSDLC Status to Deployed and close stories and bugs
    - If issues identified with any items, ensure appropriate bugs/ tasks are created to reflect these
    - Confirm all bugs that require immediate resolution are resolved
    - Confirm release success via email
    - Identify any additions or changes to release notes needed based on any issues identified with the release items
- Release notes sent

Retro

- The retrospective, held the day after release, provides an opportunity to reflect on what went well, what didn't go so well, and what we can change for the future to smooth processes and work more efficiently
- Retro boards are created and shared with the team in advance; they can also be found in the Retrospectives section of DevOps
    - The link to the retro board for the sprint should be sent to the team as early as possible to allow them to add reflections throughout the sprints
- During the retro, the first 5-10 minutes are spent adding feedback items.
- These are then talked through and grouped into similar themes
- Any additional feedback points or ideas that arise during discussion should be captured
- The team should then vote on the ideas and feedback points they find most impactful
- It's agreed during the retro which suggestion(s) be implemented and who will be responsible for taking them forwards
- User stories are then created for each agreed item and assigned appropriately

Process notes

General

Prioritisation and new stories

- All new stories should be parented to the Triage feature until confirmed will do. They should then be parented to the appropriate feature – either a backlog feature, if date undetermined, or one of the quarterly features if expecting to start within a specific quarter. If in the current or next quarter, they should be assigned to a sprint.
    - For more detail, please refer to WIN Change & Enhancement Request V1.vsdx
    - Complete analysis is not required to determine whether a story is will do/ won't do. However, the user story template should be completed as this provides the information to determine will/ won't progress.
- At creation, all stories should be related to one or more 'object' features, depending on what areas the story affects, so we can determine e.g. for meeting summaries, all of the changes we've made relating to these.
- If a new object needs creating in WIN for any development, an additional object feature should be created.
- Some of the features may be referred to as Proxy Objects, and may not be actual objects in WIN e.g. Change Management.
- When a new user story is confirmed will-do, PO will prioritise it appropriately by dragging and dropping in the backlog view, then update the prioritisation for the sprint as well if it's been allocated to a sprint

Assignation and updates

- Stories are usually assigned to the same colleague who has an active task assigned
- There may be more than one active task on a story. In this scenario, the story should be assigned to the colleague primarily responsible for driving the story as a whole forwards.

- Responsibility for updates
    - The colleague that a story is assigned to is responsible for keeping the story up to date
    - The colleague that a task is assigned to is responsible for keeping the task up to date

Naming and content conventions

- Naming of stories broadly follow the format '[Object] – [Activity] – [Description of end state]'
    - Objects may be true SF objects or proxy objects
    - Activity will fall under the categories:
        - Analysis
        - Analysis & Build
        - Build
        - Data
- Naming of tasks
    - The generic tasks requiring creation for each (non change) story are:
        - Analysis
        - Solution
        - Build
        - QA Design
        - QA Execution
        - UAT
    - Some stories may also require tasks for and titled:
        - Data
- Separate stories are created for agreed follow-up actions from a retro. These should begin 'Retro', be linked to the 'Defects and Feedback' object feature, and be tagged Retro.
- Future dating
    - For items where no aspects of the story can be progressed until a dependency is completed, date the **story** title using '[mm/dd]'
    - Tasks should not be forward dated
- User Story content – user stories should contain the below headings to ensure sufficient information capture. These can be found by applying the appropriate template – Analysis, Analysis & Build, or Build.
    - **User Story:** As a [who], I want to [what], so that [why].
        - If appropriate, there may be multiple user stories in the description for various personas, and/or a build user story in the same description as an analysis user story.
    - **Solution:** [add *who* will recommend the best approach]
    - **Questions/Approval:** [List who to contact if there are questions, and/or who will need to approve.]
    - **Demo:** [List the non-WIN team stakeholders for demo, and identify whether they are required or optional.]
    - **Additional UAT**: [If anyone *other* than the person assigned the UAT task needs to be involved in UAT, identify them here]
        - Required for stories with a build component only (i.e. Analysis & Build or Build

        - A separate task should be created for any additional (non-WIN team) UAT
    - **Notes:** [Add anything helpful or relevant]
- Placeholder identification
    - Stories that are created as placeholders only should have [Placeholder] added at the beginning of the story title
    - These should be created where we know there will be a follow-on task from an existing story, but we don't know the detail of it yet, and we need to factor the work required in for capacity/ sprint planning
    - The predecessor user story should have a task for updating the placeholder story
    - This is typically when we split analysis and build, and the build content is dependent on the analysis
- Change management naming and content – tasks
    - Change management tasks should begin 'CM –' and have the appropriate change management task template applied
    - The generic titles for change tasks are:
        - CM - Communication
        - CM - Create QRG
        - CM - Update QRG
        - CM – Update [x] in WIN (WalkMe) and SharePoint
- Change management naming and content – stories
    - Change management stories should begin 'CM –' and have the appropriate change management user story template applied for Comms stories. For other stories, the generic analysis template can be used.
    - Change management 'Activities', rather than Analysis, Build, Data etc., will fall under:
        - Communication
        - Training
        - Engagement
        - Data (less frequent)
- WalkMe naming & process – for WalkMe templates, key items for inclusion, process etc. please refer to appropriate documents which detail this.
    - There are separate scenarios for a) a WalkMe user story that is standalone, b) a WalkMe task that is part of a WalkMe user story, and c) a WalkMe task that forms part of a development user story.

Statuses and AJGSDLC Status

- The State is updated throughout the lifecycle of a story, task, or bug at key events
    - New – not actively being worked on. Items may be moved back from later stages to New if priorities change.
    - Active – actively being worked on for analysis, development, or testing.
    - Resolved – All expected development and testing work is completed. The story has been assigned for UAT, or UAT is already complete.
    - Removed – The work item is no longer needed but may be reinstated in the future
    - Closed – The change has been completed and verified in the required environment(s) (Prod, Stage, QA)

- The AJGSDLC Status reflects progress though the SDLC and is used for stories and bugs only
    - A) Backlog = Not started
    - B) Analysis = In Analysis. Should not be moved to analysis if the tag Ready for Analysis is not present.
    - C) In Dev = In Development. Should not be moved to development if Build AC and the tag Ready for Build are not present.
    - D) On Hold: Pending Business or E) On Hold: Pending Vendor = Stories may be placed on hold when we cannot progress any aspect of them without receiving further information or similar.
    - F) Ready for QA = Development complete and changes deployed to the QA environment
    - G) In QA = In QA
    - H) Ready for UAT = QA complete and changes deployed to the Stage environment. Used alongside tag QA Commit.
    - I) In UAT = In UAT
    - J) UAT Complete = UAT complete and all bugs resolved
    - K) Schedule for Deploy = UAT documentation reviewed, behaviour as expected, and we are intending to release this in the next release. Used alongside tag Commit.
    - L) Deployed = released and verified. Used alongside State Closed.

Duplicate Management

- Identify stories, tasks, and bugs of duplicates of others by:
    - Adding the 'Duplicate Of' related type link
    - Add '(Duplicate of [surviving work item ID)' at the beginning of the work item name that will not continue to be used
    - Set the status of this work item to Removed
- Ensure required information is copied from the duplicate item onto the 'surviving' item so it contains full details

Tags

- Tags are added and maintained constantly and should be added to user stories and bugs.
- If there's a tag for UAT, Training, Communication, or Change, there should be a corresponding task for this parented to the story.
- Key tags:
    - Retro = identified during a retro and agreed as a follow up actions
    - UAT = requires UAT
    - Demo = requires any form of demo, stakeholder, team, or change management
    - Change = any form of change management associated that ISN'T training or comms
    - Training = requires any form of training, including updates to existing training materials or content
    - Communication = requires a separate communication in addition to release notes

    - Ready for Analysis = User story persona, need, and reasoning are correct and analysis can begin.
    - Ready for Dev = Analysis has been reviewed and is sufficient to proceed. Solution has been identified and, if required, reviews/ agreed it meets the documented analysis need. Build AC have been added.
    - Data = Requires a data load, change, or maintenance aspect
    - WalkMe (note: see also the WalkMe processes) = Requires a WalkMe change, update, or addition
    - Steady State = Can be done Steady State AND does not need to be aligned to a release date for messaging purposes
    - Non-SS = Requires a release, OR despite being possible to do SS requires alignment to release dates for communications etc. Steady State and Non-SS should not be used on the same story.
    - Carry over = we are not expecting to complete all required activities for the story this sprint, but will continue work on it during the sprint
    - QA Commit = QA is complete successfully
    - Commit = QA and UAT are complete and we are intending to include this story or bug in the upcoming release. Commit and Carry over should not be used on the same story.
    - Release Notes = needs communicating to end users and should be included in the release notes
    - RN Written = Release notes for this item have been written. If any end-user impacting changes occur to a story with this tag, change should be informed so they can update release notes.

Steady State

- Steady State changes can be made outside of the typical release cycle. It may be possible to do a change Steady State (SS) technically, but the change still be made as part of the release if there's a need to align comms or release notes timing
- Technically possible SS items are those that don't require code or code changes
- We don't use the Steady State iteration or feature. The iteration remains for consistency and for historical items.
- SS items are indicated through tags