

**KAISAHAN NG MAGULANG AT ANAK NA MAY KAPANSANAN (KAISAKA) INC.**

Product Backlog and Sprint Plans

| **Team Number** | 1 |
| --- | --- |
| **Section** | S16 |
| **Team Members** | Amon, Mikaela Nicole B. Benigno, James Khalel C. Campo, Roan Cedric V. Corpuz, Thara Mae P.  Chua, Gideon M. Dellosa, Mariella Jeanne A. Helbling, John Patrick P. Rivera, Paolo Eugene M. Uy, Bea Antoinette C. |
| **Date Update** | August 5, 2025 |

**Table of Contents**

[**1 Product Backlog 2**](#_bsdfmbomy64e)

[US 1.1 3](#_197ztvo5ykye)

[US 1.2 4](#_i840wlmum61y)

[US 1.3 5](#_b6pt6fqcmg13)

[US 1.4 6](#_yhd3h6ak3spb)

[US 1.5 8](#_97f59n31j444)

[US 1.6 9](#_7g1qkwke5uis)

[US 1.7 10](#_ys1a91r011fp)

[US 1.8 11](#_ugujnyshbutl)

[US 1.9 12](#_ktsqhxylo8pg)

[US 1.10 13](#_1l93wvlho6m5)

[US 1.11 15](#_c1g96m92v5uf)

[US 1.12 16](#_n43ardyl6jmu)

[US 1.13 17](#_hqjpj8z2yb6c)

[US 1.14 18](#_3s8swhe6g44c)

[US 1.15 19](#_yvagejk5mfh8)

[US 1.16 21](#_p2eyyqdweq31)

[US 1.17 22](#_hw5d7e8vlx8l)

[US 1.18 23](#_9mzhpc1bxjzc)

[US 1.19 24](#_xmqisg7mi2l7)

[US 1.20 25](#_k8yi8isbn7n0)

[US 2.1 26](#_g067nrje1jji)

[US 2.2 28](#_4h4eqe4oj8zx)

[US 2.3 29](#_3iusf8w96a8y)

[US 2.4 30](#_nr00z4cjztg)

[US 2.5 31](#_n29uzliifsq8)

[US 2.6 32](#_56tn9zda17io)

[US 2.7 34](#_w0j5keg3gy1m)

[US 2.8 35](#_dole7yd3kzfq)

[US 2.9 36](#_rkr0czakiyga)

[US 2.10 37](#_hnlvbm8dex60)

[US 2.11 38](#_5eb0vc6657ip)

[US 2.12 40](#_o2qiqffyj15g)

[US 2.13 41](#_wpij551ovjm3)

[US 2.14 42](#_n3nswvr0gv3k)

[US 2.15 43](#_nswvn9c1kljk)

[US 2.16 44](#_tgs20d5t8yl0)

[US 2.17 46](#_zz31yg75w8o)

[US 2.18 47](#_n5wzmk4zgfr9)

[US 2.19 48](#_9fiw9x4dyhds)

[US 2.20 49](#_abzg4p3y9h9a)

[US 2.21 50](#_r3d3fu5lfhsg)

[US 2.22 52](#_u1n4q15eka7u)

[US 3.1 53](#_ioyu78ywesuv)

[US 3.2 54](#_ewvx6qtp5lez)

[US 3.3 55](#_7vo3kl5ajcej)

[US 3.4 56](#_a0329duiwac)

[US 3.5 57](#_8kc63urs5srv)

[US 3.6 59](#_118jl8mf75lc)

[US 3.7 60](#_yeqeiwswsj6j)

[US 4.1 61](#_vqaiuv1bsef4)

[US 4.2 62](#_n2pxlp788boq)

[US 4.3 63](#_9hc4ey5bjgm0)

[US 4.4 65](#_ahh4h14mwi8g)

[US 4.5 66](#_qaqcphc2g7uz)

[US 4.6 67](#_igfio938mks1)

[US 4.7 68](#_e5yclz1yn0d3)

[US 4.8 69](#_ph22etfv4lk5)

[US 4.9 71](#_mc9sbmocbta0)

[US 4.10 72](#_n0mjpyue3ob)

[US 4.11 73](#_l2jgti9wwa2x)

[US 4.12 74](#_rbmm3aermtcz)

[**2 Sprint Plans 75**](#_ut1xo63lj3b1)

# Product Backlog

*This section lists down all the user stories that the team commits to for the course project and the specific tasks to accomplish them.*

## US 1.1

**As a developer, I want to prepare the team's GitHub repository with design patterns, coding standards, and boilerplate code to ensure consistency and make it easy for members to build upon.**

**Story Points:** 3  
Priority: 60  
Taken by: Roan, Gideon, Mika

**Pre-condition:** There is no central repository with standardized patterns or base files

**Scenario:** The developers set up the GitHub repo, include guidelines and folder structure, and confirm it's ready for use by the whole team

**Post-condition:** A GitHub repository exists with defined structure, documented standards, and reusable starter code

**Acceptance Criteria:**

* Repository is accessible to all members
* Boilerplate code includes basic project structure
* README contains coding conventions and folder guide
* Git commits follow a standard format (e.g., Conventional Commits)
* Successfully reviewed and acknowledged by designers and QA

**Tasks:**

1. Set up public GitHub repository with folders for backend, frontend, and documentation – 2h – Roan, Gideon, Mika
2. Write contribution guidelines and coding conventions in README – 1.5h – Roan, Gideon, Mika
3. Create basic project boilerplate with environment config and dependencies – 2h – Roan, Gideon, Mika
4. Add sample routes/controllers for backend and frontend folders – 1.5h – Roan, Gideon, Mika
5. Set up .gitignore, formatting tools (e.g., Prettier), and issue labels – 1h – Roan, Gideon, Mika

## US 1.2

**As a developer, I want to set up our database using PostgreSQL with Supabase to streamline schema design and enable the team to perform database operations more efficiently.**

**Story Points:** 5  
Priority: 70  
Taken by: Roan, Gideon, Mika

**Pre-condition:** No database schema or service has been initialized

**Scenario:** Developers configure Supabase, define schema structure, and test access by connecting with initial endpoints

**Post-condition:** Database is set up with tables for caregivers, children, families, and users, and is accessible through Supabase

**Acceptance Criteria:**

* Supabase project is active and linked to version control
* At least four initial tables are created
* All tables use correct data types and foreign key relationships
* Sample data can be queried from the backend
* Designers are able to reference schema for UI development

**Tasks:**

1. Create Supabase project and set up PostgreSQL instance – 1.5h – Roan, Gideon, Mika
2. Design and build schema for caregivers, children, users, and families – 2h – Roan, Gideon, Mika
3. Configure API keys and access restrictions for Supabase – 1h – Roan, Gideon, Mika
4. Populate tables with seed data for development – 1.5h – Roan, Gideon, Mika
5. Share schema diagram and database guide to design and QA team – 1h – Roan, Gideon, Mika

## US 1.3

**As a quality assurance developer, I want to set up test scripts and unit test boilerplate code so the QA team can efficiently verify the correctness and reliability of individual functions.**

**Story Points:** 3  
Priority: 65  
Taken by: JP, James

**Pre-condition:** There are no test cases or frameworks for automated testing

**Scenario:** QA team configures test environment, prepares testing boilerplate, and confirms integration with the repo

**Post-condition:** A testing structure is added to the repo, ready for future test scripts

**Acceptance Criteria:**

* Unit test structure is in place using a standard framework (e.g., JUnit, Jest)
* Example test case for a basic function or route included
* Test results can be viewed in console or CI logs
* QA team can clone, test, and edit scripts without error
* Designers and developers can use same test structure

**Tasks:**

1. Set up testing framework for backend and frontend (e.g., JUnit, Jest) – 2h – JP, James
2. Write sample test case for one backend and one frontend component – 1.5h – JP, James
3. Create test script for executing all unit tests in batch – 1h – JP, James
4. Document how to write and run tests in CONTRIBUTING.md – 1h – JP, James
5. Sync with developers to validate compatibility of boilerplate – 1h – JP, James

## US 1.4

**As a staff member, I want the home page to prominently feature the organization’s official logo to establish visual trust and clearly signal that I’m in the right place.**

**Story Points:** 2  
Priority: 55  
Taken by: Bea, Paolo

**Pre-condition:** Homepage lacks any organization-specific branding

**Scenario:** Designers add the logo to a header element and verify visual alignment with the KAISAKA brand

**Post-condition:** The homepage design includes the KAISAKA logo clearly visible on desktop layout

**Acceptance Criteria:**

* Official logo is centered or top-left on the homepage
* Logo complies with color and sizing guidelines
* Logo displays properly on standard desktop screen sizes
* Logo component is exported and reusable
* Designers confirm alignment with final wireframes

**Tasks:**

1. Import and position KAISAKA logo on desktop header – 1h – Bea, Paolo
2. Align spacing and size with official style guide – 1h – Bea, Paolo
3. Add layout constraints to prevent overflow or stretch – 1h – Bea, Paolo
4. Export component and prepare for reuse in other pages – 1h – Bea, Paolo
5. Annotate design file with logo rules and branding note – 0.5h – Bea, Paolo

## US 1.5

**As a staff member, I want a clearly styled login button on the home page so I can immediately access the system without confusion.**

**Story Points:** 2  
Priority: 50  
Taken by: Bea, Paolo

**Pre-condition:** Homepage does not have any call-to-action that leads to the login screen

**Scenario:** Designers place a login button using final layout and styling, verify color contrast, and prepare for frontend linking

**Post-condition:** A visually prominent login button appears on the homepage based on the wireframe

**Acceptance Criteria:**

* Button is visible without scrolling
* Contrast and color are accessible (WCAG compliant)
* Button has hover and click states
* Click leads to correct login route or modal
* Designers finalize placement in prototype and assets

**Tasks:**

1. Design and style login button on the homepage header – 1.5h – Bea, Paolo
2. Define interaction states (hover, active) – 1h – Bea, Paolo
3. Position button near or alongside the registration button – 0.5h – Bea, Paolo
4. Finalize colors, border radius, and sizing based on design system – 1h – Bea, Paolo
5. Export component for use in landing page – 1h – Bea, Paolo

## US 1.6

**As a staff member, I want a visually distinct registration button for caregivers to make the registration process easy to find and initiate.**

**Story Points:** 3  
Priority: 75  
Taken by: Bea, Paolo

**Pre-condition:** The homepage exists but lacks a clearly styled registration button for caregivers

**Scenario:** Designers add a visually distinct button → Caregivers and staff can immediately identify where to initiate the registration process

**Post-condition:** A styled caregiver registration button is visible and highlighted on the homepage design

**Acceptance Criteria:**

* Button is visually distinguishable from the login button
* Button is labeled clearly (e.g., “Register as Caregiver”)
* Button is placed prominently and passes accessibility contrast guidelines
* Final design is implemented in the wireframe and matches frontend build

**Tasks:**

1. Design and position caregiver registration button near login section – 1h – Bea, Paolo
2. Ensure button follows brand colors and maintains contrast – 1h – Bea, Paolo
3. Add hover and focus styling for interactivity – 1h – Bea, Paolo
4. Place button inside header or landing card section and align with UX flow – 1h – Bea, Paolo
5. Export button component specs and styling notes for implementation – 1h – Bea, Paolo

## US 1.7

**As a staff member, I want a minimal, uncluttered account registration form that collects only essential information, so the process feels focused and purposeful.**

**Story Points:** 5  
Priority: 85  
Taken by: Bea, Paolo

**Pre-condition:** There is no UI that collects essential account data from staff or caregivers

**Scenario:** Designers create a simplified form → Staff or caregivers can register without confusion or excessive input

**Post-condition:** A wireframed registration form layout is available, focusing only on core data

**Acceptance Criteria:**

* Only required fields (e.g., name, email, password) are displayed
* Form has grouped fields with proper labels and spacing
* Includes basic validation cues (e.g., required indicators)
* Follows standard input alignment conventions for desktop UI

**Tasks:**

1. Create wireframe for registration form with grouped sections – 2h – Bea, Paolo
2. Decide on essential input fields and exclude redundant ones – 1h – Bea, Paolo
3. Add field hints or validation indicators (e.g., asterisk for required) – 1h – Bea, Paolo
4. Design success and error message placements for submission – 1h – Bea, Paolo
5. Export form layout and annotations to guide frontend work – 1h – Bea, Paolo

## US 1.8

**As a staff member, I want a clean and straightforward login form that makes it easy to input credentials quickly and confidently.**

**Story Points:** 3  
Priority: 80  
Taken by: Bea, Paolo

**Pre-condition:** There is no existing visual design or layout for the login form

**Scenario:** Designers build a clean login form → Staff can quickly enter credentials without cognitive load

**Post-condition:** A login screen is wireframed with proper field design and user flow

**Acceptance Criteria:**

* Email and password fields are vertically aligned and labeled
* Form includes a login button and optional “forgot password” link
* UI accommodates common error messages and focus styling
* Designed form follows desktop-first layout conventions

**Tasks:**

1. Draft login form with minimalist layout and clear field labeling – 1h – Bea, Paolo
2. Add “forgot password” and basic validation indicators – 1h – Bea, Paolo
3. Ensure accessibility (e.g., focus borders, label associations) – 1h – Bea, Paolo
4. Include visual treatment for failed login states – 1h – Bea, Paolo
5. Finalize and export annotated form design – 1h – Bea, Paolo

## US 1.9

**As a staff member, I want an intuitive, well-labeled navigation bar that helps me move between pages without needing to think twice.**

**Story Points:** 5  
Priority: 85  
Taken by: Bea, Paolo

**Pre-condition:** Users do not have a persistent or well-organized navigation UI

**Scenario:** Designers implement a labeled navigation bar → Staff members can access main sections from any page

**Post-condition:** A navigation bar is created in the design with appropriate labels and layout

**Acceptance Criteria:**

* Includes clearly labeled links to core pages (e.g., Home, CYWDs, Families, Events)
* Stays consistent across pages
* Responsive to user’s login status (show/hide links accordingly)
* Visually clean and aligned with organization’s style

**Tasks:**

1. Wireframe a top navigation bar with desktop layout – 1.5h – Bea, Paolo
2. Define navigation links and label wording – 1h – Bea, Paolo
3. Add hover/focus styling for desktop mouse navigation – 1h – Bea, Paolo
4. Design a role-based visibility toggle (e.g., staff only sections) – 1.5h – Bea, Paolo
5. Export and document nav structure and interactions – 1h – Bea, Paolo

## US 1.10

**As a staff member, I want a clearly structured, easy-to-scan list of children so I can quickly locate and view individual records.**

**Story Points:** 5  
Priority: 90  
Taken by: Bea, Paolo

**Pre-condition:** Staff users don’t have an interface to see or browse the list of children

**Scenario:** Designers build a list view for children with name, age, and tags → Staff can locate records efficiently

**Post-condition:** The child list UI is complete and ready for frontend development

**Acceptance Criteria:**

* List displays child name, age, status, and key tags
* Visually clean with alternating row styling or cards
* Includes search or filter field at the top
* Prototype works in desktop-first layout

**Tasks:**

1. Design scrollable list layout with cards or rows for each child – 1.5h – Bea, Paolo
2. Add child attributes to each row (name, age, etc.) – 1h – Bea, Paolo
3. Include optional indicators (e.g., with intervention plan) – 1.5h – Bea, Paolo
4. Design search/filter bar to refine child list – 1.5h – Bea, Paolo
5. Finalize and export list view UI for dev reference – 1h – Bea, Paolo

## US 1.11

**As a staff member, I want a neatly organized list of caregivers with consistent formatting to simplify browsing and lookup.**

**Story Points:** 5  
Priority: 85  
Taken by: Bea, Paolo

**Pre-condition:** No list interface exists for staff to browse or locate caregiver records

**Scenario:** Designers build a formatted list layout → Staff members can view caregiver info at a glance

**Post-condition:** A desktop-friendly caregiver list layout with consistent formatting is complete

**Acceptance Criteria:**

* List shows caregiver name, age, and relevant tags
* Each row or card follows consistent layout
* Includes search/filter functionality
* Clearly distinguishes inactive or pending caregivers

**Tasks:**

1. Wireframe layout for caregiver list with spacing and groupings – 1.5h – Bea, Paolo
2. Design individual caregiver card or row format – 1h – Bea, Paolo
3. Add indicators for pending/verified caregivers – 1h – Bea, Paolo
4. Include searchable/filterable bar above list – 1.5h – Bea, Paolo
5. Export final design with component annotations – 1h – Bea, Paolo

## US 1.12

**As a staff member, I want a structured view of families that clearly connects caregivers with their children, so family relationships are immediately understandable.**

**Story Points:** 8  
Priority: 95  
Taken by: Bea, Paolo

**Pre-condition:** No existing UI shows caregiver-child relationships

**Scenario:** Designers create a grouped interface → Staff can easily trace family members and their relations

**Post-condition:** A family list or card layout shows each family unit with member links

**Acceptance Criteria:**

* Each family grouping shows caregivers and their children together
* Uses visual hierarchy or indentation to show relationships
* Handles multiple caregivers or children in one unit
* Layout remains readable and scannable

**Tasks:**

1. Draft family unit card or expandable row design – 2h – Bea, Paolo
2. Define how to show caregiver-child mapping visually – 1.5h – Bea, Paolo
3. Add indicators for unlinked members or verification needed – 1h – Bea, Paolo
4. Include header/search for selecting a family – 1h – Bea, Paolo
5. Export interaction flow and layout for dev work – 1h – Bea, Paolo

## US 1.13

**As a staff member, I want a consolidated list of pending verifications that’s easy to scan and filter, so I can quickly see what actions are needed.**

**Story Points:** 5  
Priority: 80  
Taken by: Bea, Paolo

**Pre-condition:** Users don’t have visibility on pending verification items

**Scenario:** Designers provide a list of verification requests → Staff can act on items requiring attention

**Post-condition:** A clean, filterable list UI shows all pending verification records

**Acceptance Criteria:**

* List clearly marks what is pending and what type (e.g., caregiver ID, child record)
* Rows/cards are scannable and consistent
* Search and filter bar available
* Action button or status icon per entry

**Tasks:**

1. Design table or list of pending verification items – 1.5h – Bea, Paolo
2. Use tags/icons to denote verification type (ID, profile, etc.) – 1h – Bea, Paolo
3. Add filters for type or urgency – 1h – Bea, Paolo
4. Include hover/click interaction for more detail – 1.5h – Bea, Paolo
5. Export design spec for developer reference – 1h – Bea, Paolo

## US 1.14

**As a staff member, I want a clearly presented list of target activities, categorized and labeled, so I can easily understand event objectives at a glance.**

**Story Points:** 5  
Priority: 75  
Taken by: Bea, Paolo

**Pre-condition:** No design exists to display planned outreach activities

**Scenario:** Designers organize a list of categorized activities → Staff sees goal-oriented events and their types

**Post-condition:** Staff can view the activity list grouped by category and labeled by purpose

**Acceptance Criteria:**

* Activities are grouped (e.g., medical, educational, livelihood)
* Each entry includes date, objective, and status
* UI uses visual separation (e.g., tabs or sections)
* Design supports future status updates

**Tasks:**

1. Design categorized list of activity cards or rows – 1.5h – Bea, Paolo
2. Create icons or color codes for each activity type – 1h – Bea, Paolo
3. Include section headers or tabs for categories – 1h – Bea, Paolo
4. Add optional filters (e.g., by month or program type) – 1h – Bea, Paolo
5. Finalize UI layout and label scheme – 1h – Bea, Paolo

## US 1.15

**As a staff member, I want a dated list of completed activities with a clean, uniform layout to track progress and review past events efficiently.**

**Story Points:** 5  
Priority: 80  
Taken by: Bea, Paolo

**Pre-condition:** Past activity records are not easily viewable in the interface

**Scenario:** Designers make a history list UI → Staff can see completed events in chronological order

**Post-condition:** Completed activities are listed with dates and uniform styling

**Acceptance Criteria:**

* Each row shows event name, date, and summary
* Chronologically sorted
* Layout is consistent with target activity list
* Completed status visually marked

**Tasks:**

1. Design past event list UI with date headers or sort – 1.5h – Bea, Paolo
2. Ensure layout symmetry with target activity list – 1h – Bea, Paolo
3. Add “Completed” indicator on each card/row – 1h – Bea, Paolo
4. Include pagination or scroll design for long lists – 1h – Bea, Paolo
5. Export design with explanation of list order and tags – 1h – Bea, Paolo

## US 1.16

**As a staff member, I want an event attendance view that’s easy to navigate and visually signals attendance status, so I can assess participation at a glance.**

**Story Points:** 8  
Priority: 90  
Taken by: Bea, Paolo

**Pre-condition:** No interface exists for viewing participant attendance in events

**Scenario:** Designers create an attendance-focused layout → Staff can monitor who attended which event

**Post-condition:** An attendance view displays attendees and their statuses in an organized format

**Acceptance Criteria:**

* View shows each attendee’s name and status (present, absent, etc.)
* Attendance icons or labels are visually distinct
* Layout supports multiple events and dates
* Interface is responsive and scannable

**Tasks:**

1. Design table or list layout for attendance per event – 2h – Bea, Paolo
2. Develop a legend for attendance statuses with icons/colors – 1.5h – Bea, Paolo
3. Draft layout to show both names and role (e.g., caregiver, child) – 1.5h – Bea, Paolo
4. Add a dropdown or filter for selecting event date – 1h – Bea, Paolo
5. Export component spec for implementation – 1h – Bea, Paolo

## US 1.17

**As a staff member, I want a well-structured child registration form with grouped fields and logical flow to make data entry more focused and less error-prone.**

**Story Points:** 8  
Priority: 100  
Taken by: Bea, Paolo

**Pre-condition:** Staff have no clean interface for registering CYWDs

**Scenario:** Designers structure a form interface → Staff can register children accurately and without confusion

**Post-condition:** A grouped and step-wise child registration form is ready for implementation

**Acceptance Criteria:**

* Fields are grouped by category (basic info, conditions, etc.)
* Logical progression top-to-bottom
* Supports validation and error display
* Visual cues (e.g., required fields, tooltips) are included

**Tasks:**

1. Define form field groupings and logical layout – 2h – Bea, Paolo
2. Design page or modal for multi-step input (if needed) – 1.5h – Bea, Paolo
3. Include field labels, hints, and placeholders – 1.5h – Bea, Paolo
4. Add UI support for validation and error display – 1h – Bea, Paolo
5. Finalize prototype for registration form – 1h – Bea, Paolo

## US 1.18

**As a staff member, I want a simple, distraction-free caregiver registration form that allows me to focus on key inputs without unnecessary steps.**

**Story Points:** 5  
Priority: 90  
Taken by: Bea, Paolo

**Pre-condition:** Caregiver registration is not yet designed

**Scenario:** Designers define a clear form UI → Staff can onboard caregivers with minimal distraction

**Post-condition:** A straightforward caregiver registration form exists for frontend development

**Acceptance Criteria:**

* Only key fields (name, contact, relation) are shown
* Layout is minimal and without clutter
* Clearly shows required vs. optional fields
* Can support adding to an existing family

**Tasks:**

1. Design minimal layout with max 1–2 columns – 1.5h – Bea, Paolo
2. Set up clear visual distinction between required/optional – 1h – Bea, Paolo
3. Add field to optionally link to a family – 1h – Bea, Paolo
4. Review for visual distractions and streamline further – 1h – Bea, Paolo
5. Finalize design handoff with notes – 1h – Bea, Paolo

## US 1.19

**As a staff member, I want a dynamic checkbox that reveals a family search feature, making it easy to link new members to existing families without cluttering the default view.**

**Story Points:** 5  
Priority: 85  
Taken by: Bea, Paolo

**Pre-condition:** No dynamic field toggling exists in forms

**Scenario:** Designers add a toggle-driven UI → Staff can optionally search for and attach family data

**Post-condition:** Checkbox and search field are conditionally shown during registration

**Acceptance Criteria:**

* Checkbox labeled clearly (e.g., “Link to existing family?”)
* When checked, a search field appears
* Search UI includes suggestions or results
* Default view is clean without the search field

**Tasks:**

1. Design checkbox with conditional input field behavior – 1.5h – Bea, Paolo
2. Create interface for family search input with sample result – 1.5h – Bea, Paolo
3. Optimize for simplicity in default state – 1h – Bea, Paolo
4. Add toggle animation or smooth reveal – 1h – Bea, Paolo
5. Export design behavior as micro-interaction spec – 1h – Bea, Paolo

## US 1.20

**As a staff member, I want the system to organize children and family information clearly and accurately, so I can easily view, register, and manage family relationships and child records without confusion or data duplication.**

**Story Points:** 13  
Priority: 100  
Taken by: Bea, Paolo

**Pre-condition:** No centralized structure for viewing/managing families and children exists

**Scenario:** Designers connect registration and list UIs → Staff can manage relationships with clarity

**Post-condition:** A system of interconnected views for children and family information is designed

**Acceptance Criteria:**

* Caregiver and child lists can be accessed from each other
* Family groupings are consistent across views
* Linking during registration reflects in the data views
* Layouts prioritize accuracy and visibility

**Tasks:**

1. Design interaction flows linking child/caregiver/family views – 2.5h – Bea, Paolo
2. Review and ensure consistency between list views and registration outputs – 2h – Bea, Paolo
3. Add UI warnings for duplicated family links – 1.5h – Bea, Paolo
4. Align all visual styles across views – 1.5h – Bea, Paolo
5. Package all views and interactions for frontend implementation – 1h – Bea, Paolo

## US 2.1

**As a staff member, I want the home page to load the organization's official logo from the server and display it prominently, so I immediately recognize the platform and confirm I’m in the right place.**

**Story Points:** 3  
Priority: 95  
Taken by: Bea, Paolo, Roan, Gideon, Mika

**Pre-condition:** Logo was only a static placeholder in the frontend

**Scenario:** Designers ensure the logo is visible and prioritized → Developers load and serve the logo dynamically

**Post-condition:** Logo loads from the backend and displays clearly at the top of the homepage

**Acceptance Criteria:**

* Logo is retrieved from the backend on homepage load
* Logo is responsive and prominently placed
* Image formats are optimized (e.g., SVG or PNG)
* Backend supports logo update/replacement

**Tasks:**

1. Create frontend space and visual treatment for logo – 1h – Bea, Paolo
2. Set up backend storage and endpoint for logo – 2h – Roan, Gideon, Mika
3. Connect frontend to fetch and display logo from server – 2h – Bea, Paolo
4. Handle fallback if logo fails to load – 1h – Bea, Paolo
5. Add dynamic logo update support (admin-editable) – 2h – Roan, Gideon, Mika

## US 2.2

**As a staff member, I want a responsive login button on the home page that directs me to a secure login route, allowing quick and safe authentication.**

**Story Points:** 3  
Priority: 95  
Taken by: Bea, Paolo, Roan, Gideon, Mika

**Pre-condition:** Home page has no working login button

**Scenario:** Designers create a visible button → Developers link it to secure backend login route

**Post-condition:** Clicking login button leads to a functional login route

**Acceptance Criteria:**

* Button is visually distinct, accessible
* Button links to working login route
* Backend route is secured (HTTPS, sessions/token)
* Works on all supported screen sizes

**Tasks:**

1. Design a responsive, stylized login button – 1h – Bea, Paolo
2. Add button to homepage layout and wire up action – 1h – Bea, Paolo
3. Implement secure backend login route – 2h – Roan, Gideon, Mika
4. Connect frontend button to route with error handling – 1.5h – Bea, Paolo
5. Validate redirect and response structure – 1h – Roan, Gideon, Mika

## US 2.3

**As a staff member, I want a registration button in the landing page so that users without accounts can access the registration form.**

**Story Points:** 3  
Priority: 90  
Taken by: Bea, Paolo, Roan, Gideon, Mika

**Pre-condition:** No accessible registration route from homepage

**Scenario:** Designers add a registration button → Developers link it to the registration UI

**Post-condition:** Registration button directs to correct caregiver/child registration process

**Acceptance Criteria:**

* Button labeled clearly (e.g., “Register Now”)
* Leads to correct frontend view
* Supports caregiver or family registration flow
* Button is styled consistently with rest of homepage

**Tasks:**

1. Design and place registration button on homepage – 1h – Bea, Paolo
2. Style button for clarity and accessibility – 1h – Bea, Paolo
3. Link to frontend registration route – 1h – Bea, Paolo
4. Validate route correctness and registration page load – 1h – Roan, Gideon, Mika
5. Add conditional check for role or session – 1h – Roan, Gideon, Mika

## US 2.4

**As a staff member, I want a registration form that caregivers can access, so that they can input their family’s information themselves without the staff’s assistance.**

**Story Points:** 8  
Priority: 95  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** Caregiver registration is manually done by staff

**Scenario:** Designers build self-service form → Developers connect it to backend → QA verifies correct behavior

**Post-condition:** Caregivers can fill in and submit their data independently

**Acceptance Criteria:**

* Registration form accessible via landing page
* Validates essential fields (name, relation, etc.)
* Links data to existing or new family group
* Feedback is shown upon submission
* Data persists correctly in backend

**Tasks:**

1. Design a clean and accessible caregiver registration form – 2h – Bea, Paolo
2. Implement frontend form validation and UX – 2h – Bea, Paolo
3. Build backend handler to save caregiver data – 2h – Roan, Gideon, Mika
4. Link frontend to backend and confirm persistence – 1.5h – Bea, Paolo
5. Write and run unit tests on registration submission – 2h – JP, James

## US 2.5

**As a staff member, I want a login form that securely validates credentials against the backend and clearly notifies me of any login issues.**

**Story Points:** 8  
Priority: 100  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** Login UI is not connected to authentication backend

**Scenario:** Designers finalize login form → Developers implement backend validation → QA tests credential flows

**Post-condition:** Staff and caregivers can log in and receive real-time feedback

**Acceptance Criteria:**

* Validates credentials securely over HTTPS
* Displays clear error messages (e.g., “Invalid password”)
* Redirects users to dashboard upon success
* Does not expose sensitive info in frontend

**Tasks:**

1. Finalize login form layout and design – 1.5h – Bea, Paolo
2. Implement frontend field validation and alert handling – 1.5h – Bea, Paolo
3. Implement backend login route with credential validation – 2.5h – Roan, Gideon, Mika
4. Link frontend form to backend with error display – 1.5h – Bea, Paolo
5. Test login flow and edge cases with unit tests – 2h – JP, James

## US 2.6

**As a staff member, I want a navigation bar that dynamically adapts based on my session or role, allowing seamless transitions between authorized sections.**

**Story Points:** 8  
Priority: 98  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** The site uses a static navigation bar for all users

**Scenario:** Designers prepare a dynamic navbar layout → Developers implement logic based on role/session → QA validates access and routing

**Post-condition:** Logged-in users see navigation items based on their roles

**Acceptance Criteria:**

* Navbar shows relevant links depending on login status and role
* Non-authenticated users see limited items (e.g., login, register)
* Authenticated staff/caregivers see full authorized menu
* Navigation buttons lead to correct pages and update correctly on logout/login

**Tasks:**

1. Design dynamic navbar UI for various roles – 1.5h – Bea, Paolo
2. Implement conditional logic for rendering nav items – 2h – Bea, Paolo
3. Create backend session/role-checking endpoints – 2.5h – Roan, Gideon, Mika
4. Link frontend nav to session/role state – 1.5h – Bea, Paolo
5. Write tests to check correct role-based navbar behavior – 2h – JP, James

## US 2.7

**As a staff member, I want a CYWD list view that retrieves up-to-date data from the backend and presents it in a searchable format so that I can easily browse CYWDs.**

**Story Points:** 8  
Priority: 98  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** No structured list of CYWDs is viewable from the frontend

**Scenario:** Designers build layout → Backend fetches records → QA ensures all records load and searching works

**Post-condition:** Staff can browse and search through a real-time list of CYWDs

**Acceptance Criteria:**

* List loads CYWDs from backend
* Includes full name, age, sex, and status
* Search functionality filters results live
* Search handles partial matches

**Tasks:**

1. Design structured and accessible CYWD list UI – 2h – Bea, Paolo
2. Build frontend table and live search bar – 2h – Bea, Paolo
3. Implement backend route to fetch CYWDs – 2.5h – Roan, Gideon, Mika
4. Connect frontend to backend API – 1.5h – Bea, Paolo
5. Test full list loading, filtering, and error handling – 2h – JP, James

## US 2.8

**As a staff member, I want a caregiver list view that retrieves up-to-date data from the backend and presents it in a searchable format so that I can easily browse caregivers.**

**Story Points:** 8  
Priority: 96  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** Caregivers must be manually searched in raw data

**Scenario:** Designers design the list format → Developers connect backend → QA tests search/filter functions

**Post-condition:** Staff can easily view and search through all registered caregivers

**Acceptance Criteria:**

* List includes caregiver name, contact, and family link
* Search bar filters by name or family
* Data is pulled dynamically from backend
* Empty states are handled properly

**Tasks:**

1. Design caregiver list table layout and style – 2h – Bea, Paolo
2. Build search and filter frontend logic – 1.5h – Bea, Paolo
3. Implement caregiver list API from backend – 2.5h – Roan, Gideon, Mika
4. Integrate list view with API and handle loading/errors – 1.5h – Bea, Paolo
5. Run QA tests for list rendering, edge cases – 2h – JP, James

## US 2.9

**As a staff member, I want a family list view that displays caregivers and their associated children based on backend-linked relationships, so I can easily understand family structures.**

**Story Points:** 8  
Priority: 96  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** There is no clear family-level view connecting members

**Scenario:** Designers plan layout → Developers fetch and map data → QA validates relationship accuracy

**Post-condition:** Staff can see each family with its caregivers and children in one view

**Acceptance Criteria:**

* List groups caregivers with linked CYWDs
* Backend provides joined data per family unit
* Expandable/collapsible list items if needed
* Clear labels and identifiers for each individual

**Tasks:**

1. Design nested family list layout – 2h – Bea, Paolo
2. Build UI for expandable family rows – 2h – Bea, Paolo
3. Implement backend endpoint joining caregivers and CYWDs – 2.5h – Roan, Gideon, Mika
4. Connect frontend to new backend data format – 1.5h – Bea, Paolo
5. Verify proper linkage and display per family – 2h – JP, James

## US 2.10

**As a staff member, I want a pending verifications list that fetches real-time document verification statuses from the server and lets me track, filter, and act on incomplete records.**

**Story Points:** 8  
Priority: 94  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** Verification tracking is done manually or offline

**Scenario:** Designers create a verification list UI → Developers fetch statuses from backend → QA checks interaction and filtering

**Post-condition:** Staff can monitor which families or users still have incomplete or unverified records

**Acceptance Criteria:**

* List loads entries needing verification
* Includes filters (e.g., type of doc, status)
* Actions like “Mark as Verified” or “Request Update” available
* Reflects changes instantly after action

**Tasks:**

1. Design UI for verification status display – 2h – Bea, Paolo
2. Build real-time filter and action buttons – 1.5h – Bea, Paolo
3. Implement backend status query and update logic – 2.5h – Roan, Gideon, Mika
4. Connect UI and sync with backend state – 1.5h – Bea, Paolo
5. Test full flow of verification filtering and updates – 2h – JP, James

## US 2.11

**As a staff member, I want a target activities list populated from the backend that lets me view upcoming goals or events, organized by date or priority.**

**Story Points:** 5  
Priority: 90  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** No existing interface for viewing planned activities

**Scenario:** Designers map out the activity list UI → Developers connect with backend activities → QA tests list sorting and rendering

**Post-condition:** Staff can view a structured list of upcoming events with correct data

**Acceptance Criteria:**

* Data loads dynamically from backend
* Shows activity name, date, priority
* Sorted by date or priority
* Loading and empty states handled

**Tasks:**

1. Design target activities list layout and states – 2h – Bea, Paolo
2. Build frontend list display with sorting options – 1.5h – Bea, Paolo
3. Develop backend endpoint for retrieving target activities – 2h – Roan, Gideon, Mika
4. Connect list view with backend and display logic – 1.5h – Bea, Paolo
5. Test data rendering, sorting, and failure cases – 1.5h – JP, James

## US 2.12

**As a staff member, I want a conducted activities view that reflects completed events with timestamps from the backend, so I can reliably track organizational progress.**

**Story Points:** 5  
Priority: 88  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** Completed events cannot be easily reviewed or retrieved

**Scenario:** Designers create UI layout for completed activities → Developers fetch completed data → QA ensures date and data accuracy

**Post-condition:** Staff can browse past events with dates and details

**Acceptance Criteria:**

* Shows a list of completed activities
* Includes date completed, description
* Pulled from backend via timestamp filtering
* Clear distinction from upcoming/ongoing

**Tasks:**

1. Design completed activities list interface – 2h – Bea, Paolo
2. Build UI for displaying past events – 1.5h – Bea, Paolo
3. Implement filtering logic in backend for completed activities – 2h – Roan, Gideon, Mika
4. Sync frontend display with backend filters – 1.5h – Bea, Paolo
5. Test correct data loading and edge date cases – 1.5h – JP, James

## US 2.13

**As a staff member, I want each event’s attendance list to load with accurate status indicators per attendee, pulled directly from backend records for consistency.**

**Story Points:** 8  
Priority: 95  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** Attendance status is not visible or updated in real time

**Scenario:** Designers outline layout for attendance per event → Backend returns per-attendee status → QA checks accuracy and completeness

**Post-condition:** Attendance records are viewable per event with correct status

**Acceptance Criteria:**

* List shows each attendee for an event
* Status labels (e.g., Present, Absent) included
* Reflects real-time status from backend
* Accessible and scrollable interface

**Tasks:**

1. Design per-event attendance interface with status colors – 2h – Bea, Paolo
2. Build frontend list for attendee details and status – 2h – Bea, Paolo
3. Backend: fetch attendee list with attendance statuses – 2.5h – Roan, Gideon, Mika
4. Integrate attendance records into event details page – 1.5h – Bea, Paolo
5. Validate status rendering and real-time changes – 2h – JP, James

## US 2.14

**As a staff member, I want the child registration form to submit clean, validated data to the backend and return feedback if anything is missing or invalid.**

**Story Points:** 8  
Priority: 99  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** Child registration data is not validated or doesn't reach backend properly

**Scenario:** Designers format a clear form → Developers handle submission and validation → QA checks all edge cases and errors

**Post-condition:** Staff can submit child info with form validation and backend integration

**Acceptance Criteria:**

* Fields are logically grouped and required fields checked
* Form data submitted to backend via API
* Invalid inputs trigger UI feedback
* Success/failure clearly shown

**Tasks:**

1. Design full child registration form UI and field grouping – 2h – Bea, Paolo
2. Add frontend validation logic (required, format, etc.) – 2h – Bea, Paolo
3. Build backend endpoint for saving child records – 2.5h – Roan, Gideon, Mika
4. Connect form to backend, return response messages – 1.5h – Bea, Paolo
5. Write form tests for empty/invalid/valid cases – 2h – JP, James

## US 2.15

**As a staff member, I want the caregiver registration form to follow a simple structure that only sends the core data needed to register a new caregiver into the system.**

**Story Points:** 5  
Priority: 92  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** There is no form or only a placeholder for registering caregivers

**Scenario:** Designers simplify the form → Backend accepts and stores caregiver info → QA tests inputs

**Post-condition:** A caregiver can be registered via a short, functional form

**Acceptance Criteria:**

* Form includes essential fields only (name, contact, etc.)
* Connected to backend via API
* Invalid inputs are handled
* Success alert or redirection after submission

**Tasks:**

1. Design minimal caregiver registration form UI – 1.5h – Bea, Paolo
2. Implement form layout and basic field validations – 1.5h – Bea, Paolo
3. Build backend route for saving caregiver data – 2h – Roan, Gideon, Mika
4. Connect and test form with backend API – 1.5h – Bea, Paolo
5. QA validation for empty/malformed input handling – 1.5h – JP, James

## US 2.16

**As a staff member, I want a checkbox toggle that dynamically reveals a live family search input, letting me query the backend for existing families and link registrations accordingly.**

**Story Points:** 5  
Priority: 90  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** No existing feature to link a new caregiver or child to an existing family

**Scenario:** Staff checks a box to reveal a dynamic search field → Enters a query → Gets real-time results from backend → Selects and links a family

**Post-condition:** Staff can easily link new registrations to existing families using a search field

**Acceptance Criteria:**

* Checkbox toggles visibility of the search bar
* Search bar queries backend in real time
* Results are selectable and linkable to form
* UI feedback for selected family

**Tasks:**

1. Design toggle + dynamic search UI – 1.5h – Bea, Paolo
2. Implement real-time search bar and toggle logic – 1.5h – Bea, Paolo
3. Create backend route for live family search – 2h – Roan, Gideon, Mika
4. Connect frontend component to live search – 1.5h – Bea, Paolo
5. Test real-time query and link functionality – 1.5h – JP, James

## US 2.17

**As a staff member, I want a clean and well-organized dashboard that clearly highlights the proportion of children with intervention plans and presents visually distinct buttons for viewing lists and accessing registration forms, so I can quickly understand key information and take action without confusion.**

**Story Points:** 8  
Priority: 97  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** No dashboard exists or it lacks visual clarity and actionable elements

**Scenario:** Staff logs in and sees a dashboard → Understands intervention statistics immediately → Can navigate to lists or forms with a click

**Post-condition:** Dashboard offers key metrics and action buttons in an intuitive layout

**Acceptance Criteria:**

* Proportion of CYWDs with intervention plans is displayed
* Buttons for lists and registration are styled and labeled
* Layout is responsive and clutter-free
* Loads data on page load from backend

**Tasks:**

1. Design dashboard layout and highlight components – 2h – Bea, Paolo
2. Build dashboard UI with graphs and buttons – 2h – Bea, Paolo
3. Develop API to retrieve intervention plan proportions – 2.5h – Roan, Gideon, Mika
4. Integrate backend data and wire buttons – 1.5h – Bea, Paolo
5. QA tests data loading, button routing – 2h – JP, James

## US 2.18

**As a staff member, I want a dynamic dashboard that retrieves real-time data on children with intervention plans from the backend and displays their proportion visually (e.g., with charts or counters), along with functional buttons that let me seamlessly navigate to the child list view and registration form, so I can monitor, explore, and manage records efficiently.**

**Story Points:** 8  
Priority: 95  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** Static or inaccurate data on intervention status

**Scenario:** Staff accesses dashboard → Real-time stats are loaded from backend → Charts or counters show accurate proportions → Staff navigates to child list or form

**Post-condition:** A real-time visual dashboard helps staff monitor interventions and act on insights

**Acceptance Criteria:**

* Real-time data is fetched from backend
* Visual charts or counters reflect up-to-date figures
* Buttons work as navigation tools
* Works on first load and on refresh

**Tasks:**

1. Design chart-based dashboard section – 2h – Bea, Paolo
2. Build live dashboard widget for intervention stats – 2h – Bea, Paolo
3. Backend: fetch real-time child intervention data – 2.5h – Roan, Gideon, Mika
4. Sync dashboard with backend and validate state changes – 1.5h – Bea, Paolo
5. QA tests charts/counters loading and routing accuracy – 2h – JP, James

## US 2.19

**As a staff member, I want to easily track both planned and completed activities in the system, with each activity clearly associated with relevant children, families, or events, so I can monitor progress, manage participation, and follow up as needed.**

**Story Points:** 8  
Priority: 93  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** Staff cannot trace which activities have occurred or are upcoming, and lack links to relevant stakeholders

**Scenario:** Staff checks activity list → Can distinguish between planned and completed items → Can trace linked children/families

**Post-condition:** Activity view clearly connects events to participants and statuses

**Acceptance Criteria:**

* Shows both planned and completed activities
* Each activity linked to relevant CYWDs, caregivers
* Dates and status labels are accurate
* Easy to filter or sort by status or date

**Tasks:**

1. Design interface showing linked activities and stakeholders – 2h – Bea, Paolo
2. Build UI for dual list (planned + completed) with links – 2h – Bea, Paolo
3. Create backend support for linking events with CYWDs/families – 2.5h – Roan, Gideon, Mika
4. Display dynamic filters and association logic – 1.5h – Bea, Paolo
5. QA test correct data linking and list behavior – 2h – JP, James

## US 2.20

**As a staff member, I want a CYWD profile page so that I may view and edit their information to ensure it is accurate and up-to-date.**

**Story Points:** 5  
Priority: 92  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** There is no individual profile page for CYWDs

**Scenario:** Staff opens a CYWD’s profile → Sees structured fields → Can edit and save updates

**Post-condition:** Staff can view and edit each CYWD’s info from a dedicated page

**Acceptance Criteria:**

* Profile layout is clear and editable
* Data is fetched and updated via backend
* Save confirmation and error states shown
* Each field validated before submission

**Tasks:**

1. Design CYWD profile page UI with editable fields – 2h – Bea, Paolo
2. Implement frontend form logic and edit mode toggle – 1.5h – Bea, Paolo
3. Backend endpoints for get/update CYWD info – 2.5h – Roan, Gideon, Mika
4. Connect frontend with backend and validations – 1.5h – Bea, Paolo
5. QA tests for profile load, edit, update cases – 2h – JP, James

## US 2.21

**As a staff member, I want a Caregiver profile page so that I may view and edit their information to ensure it is accurate and up-to-date.**

**Story Points:** 5  
Priority: 90  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** There is no profile view/edit page for caregivers

**Scenario:** Staff accesses a caregiver’s profile → Reviews displayed details → Edits fields if needed → Saves updates

**Post-condition:** Staff can view and edit individual caregiver data from a profile page

**Acceptance Criteria:**

* Caregiver profile is accessible via list or search
* Fields are editable with validation
* Save function updates data through backend
* Confirmation and error messages shown

**Tasks:**

1. Design editable caregiver profile layout – 2h – Bea, Paolo
2. Develop frontend logic for editable fields and save – 2h – Bea, Paolo
3. Backend endpoints for fetching and updating caregiver data – 2.5h – Roan, Gideon, Mika
4. Link frontend to backend and validate save operations – 1.5h – Bea, Paolo
5. QA test field validations, data load, and save functionality – 2h – JP, James

## US 2.22

**As a staff member, I want my own staff profile page so that I may view and edit my account details to ensure that I have the right email address and revise other information as I please.**

**Story Points:** 5  
Priority: 88  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** Staff account details are not easily editable or viewable from a dedicated page

**Scenario:** Staff opens their profile page → Reviews account info → Edits name/email/password as needed → Saves changes

**Post-condition:** Staff can manage their profile settings independently

**Acceptance Criteria:**

* Staff profile page is personalized and secured
* Only own info is editable
* Save function validates and updates data
* Success and failure messages appear on save attempt

**Tasks:**

1. Design staff profile page layout and inputs – 1.5h – Bea, Paolo
2. Develop frontend component with editable fields – 2h – Bea, Paolo
3. Backend route for fetching and updating staff info – 2.5h – Roan, Gideon, Mika
4. Connect form to backend and handle auth context – 1.5h – Bea, Paolo
5. QA tests for profile page access, edit flow, and edge cases – 2h – JP, James

## US 3.1

**As a staff member, I want to view a list of quarterly/annual reports with each row consisting of the year (e.g., '24-'25), total target no. of CYWDs, and total actual no. of CYWDs served, so that I can have a general overview of the program's success.**

**Story Points:** 8  
Priority: 94  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** Reports are stored in the backend but not presented in a clear list format

**Scenario:** Staff navigates to reports page → Sees a list grouped by period → Compares target and actual values

**Post-condition:** Staff can understand overall progress at a glance using the report list

**Acceptance Criteria:**

* Report list displays period, target, and actual counts
* Layout is sortable or filterable by year
* Data is fetched from backend and displayed on load
* Proper visual hierarchy is followed

**Tasks:**

1. Design list layout for report period summaries – 2h – Bea, Paolo
2. Implement report list component with visual hierarchy – 2h – Bea, Paolo
3. Backend: Fetch report list with summary metrics – 2.5h – Roan, Gideon, Mika
4. Link list to backend and ensure updates load properly – 1.5h – Bea, Paolo
5. QA verify correct rendering, loading behavior, and filters – 2h – JP, James

## US 3.2

**As a staff member, in the reports list, I want to be able to search for a certain report period so that I can view and edit its details.**

**Story Points:** 5  
Priority: 87  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** Staff can view the full report list but cannot narrow results by time period

**Scenario:** Staff uses a search input → Types in a date range or year → Matches appear from the report list

**Post-condition:** Staff can easily locate specific reports for review or update

**Acceptance Criteria:**

* Search bar is placed above or beside the report list
* Typing a year/period filters the displayed reports
* Backend supports filtered search
* UX is responsive with no page reload

**Tasks:**

1. Design search input interface and placement in UI – 1h – Bea, Paolo
2. Develop frontend logic for search filter and UI update – 1.5h – Bea, Paolo
3. Backend endpoint modification for period-based queries – 2h – Roan, Gideon, Mika
4. Integrate search field with backend + debounce typing – 1.5h – Bea, Paolo
5. QA test search behavior with edge cases (e.g., invalid input) – 1.5h – JP, James

## US 3.3

**As a staff member, in the reports list, I want to be able to add a new report period so that I can add an accomplishment report for that specific period.**

**Story Points:** 5  
Priority: 89  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** There is no option to create a new report from the report list page

**Scenario:** Staff clicks “Add new period” → Form appears → Inputs filled → Saves new period

**Post-condition:** A new report period is added and appears in the list

**Acceptance Criteria:**

* "Add Report" button exists above the report list
* Form includes fields for year, target, actuals
* Validation required before saving
* Saved entry shows in real-time in report list

**Tasks:**

1. Design "Add Report" form modal and button layout – 2h – Bea, Paolo
2. Create frontend modal/form behavior and validation – 2h – Bea, Paolo
3. Backend route for creating a new report period – 2h – Roan, Gideon, Mika
4. Hook frontend form to backend creation logic – 1.5h – Bea, Paolo
5. QA test form flow, required fields, and list update – 2h – JP, James

## US 3.4

**As a staff member, in the reports list, I want an edit button to allow me to edit the details of the period’s accomplishment report.**

**Story Points:** 5  
Priority: 90  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** Staff can view report periods but cannot modify them

**Scenario:** Staff clicks "Edit" → Form shows prefilled report values → Changes values → Saves update

**Post-condition:** Report data is updated and reflected in the list

**Acceptance Criteria:**

* Each report entry has an edit button
* Editable fields are pre-populated
* Edits trigger a backend update
* Success/failure feedback displayed

**Tasks:**

1. Design edit form layout matching view mode – 1.5h – Bea, Paolo
2. Implement editable form toggle and logic – 2h – Bea, Paolo
3. Backend route for updating an existing report – 2h – Roan, Gideon, Mika
4. Connect frontend edit flow to backend with feedback – 1.5h – Bea, Paolo
5. QA test editing accuracy, data persistence, and errors – 2h – JP, James

## US 3.5

**As a staff member, for each report period in the reports list, I want an editable form so that I can view, enter, and edit the most important details of the period’s accomplishment report.**

**Story Points:** 8  
Priority: 91  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** Reports are not editable in detail beyond a general list

**Scenario:** Staff clicks a report → Full form opens with fields for targets, interventions, etc. → Edits and saves

**Post-condition:** Full report details are accessible and editable per period

**Acceptance Criteria:**

* Fields include target CYWDs, actuals, intervention notes
* Form includes section grouping for clarity
* Backend supports both retrieve and update
* Save results in confirmation with error handling

**Tasks:**

1. Design full editable report form structure – 2.5h – Bea, Paolo
2. Build frontend input components with validation – 2.5h – Bea, Paolo
3. Backend logic to fetch and update full report record – 2.5h – Roan, Gideon, Mika
4. Connect UI to backend and display load/save states – 2h – Bea, Paolo
5. QA test full form functionality and save integrity – 2h – JP, James

## US 3.6

**As a staff member, for each report period in the reports list, I want an export button so that I can export the report's data into an Excel sheet.**

**Story Points:** 8  
Priority: 93  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** Reports exist in the backend but are not exportable as a file

**Scenario:** Staff clicks "Export" → System generates and downloads Excel file → User opens file

**Post-condition:** Staff receives a well-structured report Excel file

**Acceptance Criteria:**

* Each report has an "Export to Excel" button
* Export includes correct data and formatting
* Excel includes all relevant fields/tables
* File downloads immediately with proper filename

**Tasks:**

1. Design mockup of Excel layout and tables – 1.5h – Bea, Paolo
2. Implement export button and loading state – 1.5h – Bea, Paolo
3. Backend logic to format and generate Excel file – 3h – Roan, Gideon, Mika
4. Link button to trigger backend Excel creation and download – 1.5h – Bea, Paolo
5. QA test file content, format, and file generation – 2h – JP, James

## US 3.7

**As a staff member, in the exported reports sheet, I want a template of all tables with cells left blank if their values are not auto-generated so I can choose to edit those cells manually in the sheet instead of the website.**

**Story Points:** 5  
Priority: 88  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** Staff can export data, but no templates or editable placeholders are provided for manual adjustment

**Scenario:** Staff clicks "Export" → A sheet is generated with template tables → Blank cells are present where user input is needed

**Post-condition:** The exported file allows staff to manually fill in remaining data if needed

**Acceptance Criteria:**

* Exported Excel includes labeled sections/tables
* Auto-generated fields are filled, and others left blank
* Template is consistent in format and editable
* Staff can update the file manually after download

**Tasks:**

1. Draft table layout mockup with placeholders – 1.5h – Bea, Paolo
2. Implement logic for selectively populating data – 2h – Roan, Gideon, Mika
3. Adjust Excel generation script to include blank/editable fields – 2h – Roan, Gideon, Mika
4. Test export for formatting and editability in Excel – 1.5h – JP, James
5. Polish export visuals (spacing, table headers, footnotes) – 1h – Bea, Paolo

## US 4.1

**As a staff member, in the exported sheet, I want to view the auto-generated target and actual counts of old and new CYWDs served, so that reporting is more accurate and consistent.**

**Story Points:** 8  
Priority: 97  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** CYWDs are already registered in the system, but no automated totals are available in the exported report

**Scenario:** Staff exports a report → Excel file includes calculated fields for new vs. old CYWDs → User reviews or forwards file

**Post-condition:** Report contains validated and consistent counts of old and new CYWDs, both served and targeted

**Acceptance Criteria:**

* Backend computes and inserts correct counts
* Totals distinguish between old and new CYWDs
* Display formatting is clear and sectioned in the Excel sheet
* Manual editing remains possible for designated cells

**Tasks:**

1. Design section layout for target/actual CYWDs in Excel – 1h – Bea, Paolo
2. Implement backend calculation logic for report generation – 3h – Roan, Gideon, Mika
3. Update Excel export code to inject calculated values – 2h – Roan, Gideon, Mika
4. Validate logic with mock data – 1.5h – JP, James
5. Polish exported sheet formatting – 1h – Bea, Paolo

## US 4.2

**As a staff member, in the exported sheet, I want to view the auto-generated total number of old and new children with Down Syndrome served, so that data entry errors can be minimized.**

**Story Points:** 5  
Priority: 92  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** The export report does not distinguish children with Down Syndrome or account for their specific counts

**Scenario:** Staff exports data → Excel sheet now includes dedicated rows showing total served CYWDs with Down Syndrome by age

**Post-condition:** Exported report allows staff to see the program’s specific reach for children with Down Syndrome

**Acceptance Criteria:**

* Backend computes filtered count of Down Syndrome cases
* Section for these values is clearly labeled in the Excel sheet
* Auto-generated values are not editable
* Counts align with database records

**Tasks:**

1. Design labeled row/section for Down Syndrome data in sheet – 1h – Bea, Paolo
2. Add filtering logic for DS cases during export – 2h – Roan, Gideon, Mika
3. Connect logic to the export template generator – 1.5h – Roan, Gideon, Mika
4. Validate counts and filter logic with sample DB entries – 1.5h – JP, James
5. Final UI polish and cell locking for DS count rows – 1h – Bea, Paolo

## US 4.3

**As a staff member, in the exported sheet, I want a table showing the auto-generated number of CYWDs with membership, intervention plans, transition/graduation plans, and recorded improvements, grouped by age group and gender, so that the intervention information is readily structured for analysis of the program’s overall success.**

**Story Points:** 13  
Priority: 95  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** The system contains CYWD intervention data, but no structured export view exists for grouped analysis

**Scenario:** Staff exports the report → A structured table is generated showing counts per category by age and gender

**Post-condition:** Staff sees summarized intervention data in an analysis-ready format

**Acceptance Criteria:**

* Table in the export shows membership, intervention, graduation, and improvements
* Rows are grouped by gender and age brackets
* Counts reflect real backend data
* Table is clearly labeled and formatted

**Tasks:**

1. Design grouped table layout (gender x age group x intervention category) – 2h – Bea, Paolo
2. Implement data aggregation by group and category – 3h – Roan, Gideon, Mika
3. Connect export logic to inject grouped results – 2.5h – Roan, Gideon, Mika
4. Validate export accuracy and grouping logic – 2h – JP, James
5. Polish export table with consistent spacing and legends – 1h – Bea, Paolo

## US 4.4

**As a staff member, in the exported sheet, I want the Access to Education table to display the auto-generated number of CYWDs already enrolled last school year and newly enrolled this school year, as well as those who dropped out or completed this year, so that educational trends can be easily monitored.**

**Story Points:** 8  
Priority: 91  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** Enrollment status exists in the database, but is not reflected in reports

**Scenario:** Staff exports the sheet → An education-specific table appears, showing counts of enrolled, dropped out, and graduated CYWDs

**Post-condition:** The staff can clearly view educational status transitions and trends

**Acceptance Criteria:**

* Table includes: Enrolled (last year), Newly Enrolled, Dropped Out, Completed
* CYWDs are categorized accurately based on record history
* Totals are calculated per category
* Export table is labeled and editable only where needed

**Tasks:**

1. Design education status table with clear enrollment breakdown – 1.5h – Bea, Paolo
2. Add backend queries to segment CYWDs by enrollment transitions – 2.5h – Roan, Gideon, Mika
3. Update export logic to render segmented data – 2h – Roan, Gideon, Mika
4. Verify correct counts with test cases – 1.5h – JP, James
5. Polish cell alignments and captions – 1h – Bea, Paolo

## US 4.5

**As a staff member, in the exported sheet, I want a Type of Education table with auto-generated counts of CYWDs per education type, grouped by age and gender, so that the CYWDs’ educational backgrounds are well-structured for analysis.**

**Story Points:** 8  
Priority: 89  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** Education type exists in child profiles but is not summarized for reporting

**Scenario:** Staff generates a report → A table appears showing education type breakdown by age and gender

**Post-condition:** Export contains categorized education background data for each CYWD

**Acceptance Criteria:**

* Rows are grouped by gender and age group
* Columns represent education types (e.g., SPED, ALS, etc.)
* Counts are accurate per group
* Design is visually digestible

**Tasks:**

1. Create grouped education table layout in design – 1.5h – Bea, Paolo
2. Create backend aggregation logic by education type – 2.5h – Roan, Gideon, Mika
3. Export formatted education-type table with dynamic rows – 2h – Roan, Gideon, Mika
4. Validate logic with sample education profiles – 1.5h – JP, James
5. Refine table for consistent row/column sizing – 1h – Bea, Paolo

## US 4.6

**As a staff member, in the exported sheet, I want the Access to Social Protection table to include auto-generated figures for CYWDs with access to social protection in the previous and current year, so that the number of CYWDs that have yet to access social protection can be more easily identified.**

**Story Points:** 5  
Priority: 87  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** Social protection data is recorded per year but not surfaced in reporting

**Scenario:** Staff exports report → Table shows side-by-side yearly comparisons

**Post-condition:** Exported table reflects progress and gaps in social protection access

**Acceptance Criteria:**

* Separate columns for previous and current year
* Rows show count of CYWDs with/without access
* Totals and deltas are correct
* Formatting is clear

**Tasks:**

1. Design comparison table for 2-year social protection access – 1h – Bea, Paolo
2. Implement backend logic to compare current vs previous year – 2h – Roan, Gideon, Mika
3. Output calculated counts in export file – 1.5h – Roan, Gideon, Mika
4. Test consistency between years – 1h – JP, James
5. Tweak export visual layout for comparison clarity – 1h – Bea, Paolo

## US 4.7

**As a staff member, in the exported sheet, I want the Access to Social Protection table to reflect auto-generated numbers of CYWDs participating in family life and community groups or clubs, so that their social inclusion can be more easily assessed.**

**Story Points:** 5  
Priority: 85  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** The CYWDs’ community involvement data is stored but not presented in the report

**Scenario:** Report is generated → A new section under Access to Social Protection shows community group involvement stats

**Post-condition:** Staff can assess levels of community inclusion for CYWDs

**Acceptance Criteria:**

* Export includes separate section for community involvement
* Data is grouped by age and gender
* Counts are pulled from structured activity records
* Export is readable and visually distinct

**Tasks:**

1. Layout design for social participation table – 1h – Bea, Paolo
2. Add backend queries for involvement filtering – 2h – Roan, Gideon, Mika
3. Output counts to export template – 1.5h – Roan, Gideon, Mika
4. Test data against community/group participation records – 1.5h – JP, James
5. Apply color or label highlights to rows – 1h – Bea, Paolo

## US 4.8

**As a staff member, in the exported sheet, I want the Participation of Caregivers table to include auto-generated counts of caregivers by income-generating activity type, so that livelihood support can be evaluated efficiently.**

**Story Points:** 5  
Priority: 83  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** Caregivers’ income-generating activities are stored but not reflected in exports

**Scenario:** Staff exports the report → A table displays caregivers per livelihood category

**Post-condition:** Export reflects caregiver livelihood engagement, aiding support planning

**Acceptance Criteria:**

* Export includes a table grouped by income-generating activity types
* Table counts are accurate
* Layout is readable with meaningful column labels
* Table distinguishes activities across caregivers

**Tasks:**

1. Design layout for income-generating activity table – 1h – Bea, Paolo
2. Fetch caregiver activity data from database – 2h – Roan, Gideon, Mika
3. Populate export with grouped activity counts – 1.5h – Roan, Gideon, Mika
4. Validate correct categorization and count logic – 1.5h – JP, James
5. Polish export visual spacing and column headers – 1h – Bea, Paolo

## US 4.9

**As a staff member, in the exported sheet, I want the Participation of Caregivers table to include auto-generated counts of caregivers by community group involvement, so that the caregivers’ levels of engagement is more easily identified.**

**Story Points:** 5  
Priority: 82  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** Involvement of caregivers in community groups is recorded but not exported

**Scenario:** Export is generated → Community group involvement table for caregivers is included

**Post-condition:** Staff can assess caregiver engagement in community organizations

**Acceptance Criteria:**

* Export includes table of caregiver involvement per group
* Counts match stored records
* Clear labels for types of community involvement
* Accurate group-wise totals

**Tasks:**

1. Design table to display caregiver group involvement – 1h – Bea, Paolo
2. Add query to group caregivers by group/club participation – 2h – Roan, Gideon, Mika
3. Populate export with correct grouping – 1.5h – Roan, Gideon, Mika
4. Test against various caregiver profiles – 1.5h – JP, James
5. Refine header styling and column width – 1h – Bea, Paolo

## US 4.10

**As a staff member, in the exported sheet, I want the Access to Labor Market table to show auto-generated counts of CYWDs with labor market access by gender and employment type in the previous and current year, so that employment outcomes can be easily tracked.**

**Story Points:** 8  
Priority: 90  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** CYWD employment records exist but are not compiled for trend tracking

**Scenario:** Staff exports report → Employment access data is shown by year and gender

**Post-condition:** Staff can compare employment access across years and categories

**Acceptance Criteria:**

* Table is grouped by gender and year
* Columns reflect various employment types
* Accurate counts displayed per group
* Clearly distinguishes past and current year

**Tasks:**

1. Design layout for labor market access comparison – 1.5h – Bea, Paolo
2. Write backend logic to group CYWDs by gender and employment year/type – 2.5h – Roan, Gideon, Mika
3. Inject grouped data into the export – 2h – Roan, Gideon, Mika
4. Validate data grouping for current and past year – 1.5h – JP, James
5. Finalize table styling and legend – 1h – Bea, Paolo

## US 4.11

**As a staff member, in the exported sheet, I want the Access to Labor Market table to display auto-generated numbers of CYWDs with and without Down Syndrome who were unable to work, by gender, so that labor barriers can be more clearly identified.**

**Story Points:** 5  
Priority: 88  
Taken by: Bea, Paolo, Roan, Gideon, Mika, JP, James

**Pre-condition:** CYWDs with work inability are recorded with condition tags, but not visualized in export

**Scenario:** Staff exports report → New row entries show work inaccessibility data by gender and condition

**Post-condition:** Staff can identify which groups experience higher employment barriers

**Acceptance Criteria:**

* Table includes categories: With DS, Without DS
* Grouped by gender
* Column shows count of CYWDs unable to work
* Data is export-ready and aligned

**Tasks:**

1. Create visual layout for work inaccessibility data by group – 1h – Bea, Paolo
2. Implement backend filtering by disability status and employment status – 2h – Roan, Gideon, Mika
3. Insert result rows into export – 1.5h – Roan, Gideon, Mika
4. Test grouping logic on sample data – 1.5h – JP, James
5. Format row groupings with legend/notes – 1h – Bea, Paolo

## US 4.12

**As a staff member, I want to be able to automatically export the current list of CYWDs in an Excel sheet so that I can quickly generate a reference list of CYWDs with their corresponding identification numbers.**

**Story Points:** 3  
Priority: 93  
Taken by: Roan, Gideon, Mika, JP, James

**Pre-condition:** CYWD records exist in the system but lack a quick-export reference option

**Scenario:** Staff clicks export → Excel sheet is generated with current CYWD list and IDs

**Post-condition:** CYWDs are exportable on demand with clean ID formatting

**Acceptance Criteria:**

* Exported file lists all CYWDs with system ID
* Export is triggered by user action
* File is downloadable in Excel format
* IDs match backend values

**Tasks:**

1. Build endpoint to fetch current CYWD list with IDs – 2h – Roan, Gideon, Mika
2. Implement Excel generation logic – 1.5h – Roan, Gideon, Mika
3. Enable button-triggered export action – 1h – Roan, Gideon, Mika
4. QA test for export accuracy and file integrity – 1.5h – JP, James
5. Polish column names and header row – 1h – Bea, Paolo

# Sprint **Plans**

*This section details how your team’s product backlog items will be distributed to the different sprints.*

| **User Stories** | **Sprint 1** | **Sprint 2** | **Sprint 3** |
| --- | --- | --- | --- |
| US #1.1 | √ |  |  |
| US #1.2 | √ |  |  |
| US #1.3 | √ |  |  |
| US #1.4 | √ |  |  |
| US #1.5 | √ |  |  |
| US #1.6 | √ |  |  |
| US #1.7 | √ |  |  |
| US #1.8 | √ |  |  |
| US #1.9 | √ |  |  |
| US #1.10 | √ |  |  |
| US #1.11 | √ |  |  |
| US #1.12 | √ |  |  |
| US #1.13 | √ |  |  |
| US #1.14 | √ |  |  |
| US #1.15 | √ |  |  |
| US #1.16 | √ |  |  |
| US #1.17 | √ |  |  |
| US #1.18 | √ |  |  |
| US #1.19 | √ |  |  |
| US #1.20 | √ |  |  |
| US #2.1 |  | √ |  |
| US #2.2 |  | √ |  |
| US #2.3 |  | √ |  |
| US #2.4 |  | √ |  |
| US #2.5 |  | √ |  |
| US #2.6 |  | √ |  |
| US #2.7 |  | √ |  |
| US #2.8 |  | √ |  |
| US #2.9 |  | √ |  |
| US #2.10 |  | √ |  |
| US #2.11 |  | √ |  |
| US #2.12 |  | √ |  |
| US #2.13 |  | √ |  |
| US #2.14 |  | √ |  |
| US #2.15 |  | √ |  |
| US #2.16 |  | √ |  |
| US #2.17 |  | √ |  |
| US #2.18 |  | √ |  |
| US #2.19 |  | √ |  |
| US #2.20 |  | √ |  |
| US #2.21 |  | √ |  |
| US #2.22 |  | √ |  |
| US #3.1 |  | √ |  |
| US #3.2 |  | √ |  |
| US #3.3 |  | √ |  |
| US #3.4 |  | √ |  |
| US #3.5 |  | √ |  |
| US #3.6 |  | √ |  |
| US #3.7 |  | √ |  |
| US #4.1 |  |  |  |
| US #4.2 |  |  |  |
| US #4.3 |  |  | √ |
| US #4.4 |  |  | √ |
| US #4.5 |  |  | √ |
| US #4.6 |  |  | √ |
| US #4.7 |  |  | √ |
| US #4.8 |  |  | √ |
| US #4.9 |  |  | √ |
| US #4.10 |  |  | √ |
| US #4.11 |  |  | √ |
| US #4.12 |  |  | √ |