



FAIMS3 ALPHA V0.1.0 UAT

EVALUATION REPORT

For Technical Advisory Group &
Steering Committee

License Creative Commons 4.0 International - With Attribution

Contact Brian Ballsun-Stanton
<brian@faims.edu.au>

Version 1.0.0

DOI 10.5281/zenodo.5030772

Authors Brian Ballsun-Stanton
Nuria Lorente
Nathan Reid
Shawn Ross
Penny Crook



25 June 2021

Changelog

- 0.0.1 2021-06-17 Initial draft
- 0.0.2 2021-06-24 For approval
- 1.0.0 2021-06-25 Signoff and DOI

This document is licensed under Creative Commons 4.0 International - With Attribution

Copyright 2021 Macquarie University.

This document was typeset with L^AT_EX.

Macquarie University NSW 2109

Australia

E: info@faims.edu.au

W: faims.edu.au

ABN 90 952 801 237

CRICOS Provider 00002J

Contents

1	Preamble	4
2	Objectives	5
2.1	Design Objectives	5
2.2	Test Objectives	5
3	Method	6
3.1	Approach	6
3.2	Testing Environment	6
3.3	Known Problems	6
3.4	Conduct of Tests	6
4	Results	8
4.1	CouchDB and Project structure (Part 1)	8
4.2	Observation creation, retrieval, updating, and deletion on multiple operating systems (Part 2)	8
4.3	Dynamic Database interactions (Part 3)	9
4.4	Discussion (Part 4)	10
5	Evaluation	11
6	Appendices	12
6.1	Appendix 1	12

1 Preamble

On 15 June 2021 user acceptance testing (UAT) was conducted on our [Alpha Release](#). A number of tests for the creation, retrieval, update and deletion (CRUD) of data, on on multiple operating systems in online and offline environments, were successfully executed.

This report is intended as a short summary and evaluation of the UAT.

2 Objectives

2.1 Design Objectives

FAIMS3 is a ground-up rewrite of the [FAIMS Mobile \(v2.6\)](#) offline-capable, geospatial, multimedia, field-data collection application (Ballsun-Stanton et al. 2018). This rewrite is designed to be multi-platform, maintainable, and to support data collection at a citizen-science scale. The code and platform should last for at least five years, assuming regular maintenance.

For more information on the overarching design approach see the [FAIMS3 Elaboration Report](#).

The objectives for the **Alpha release** were:

To demonstrate the foundational capabilities of FAIMS3. Specifically, loading a module from a specification, data entry on all OSes, and asynchronous data exchange on an append-only datastore.

2.2 Test Objectives

The intention of user acceptance testing is to explore, discuss, and confirm accomplishment of the goals of Alpha Development (above).

This Alpha prototype demonstrates core functionality of FAIMS3's Electronic Field Notebook platform, which is largely invisible to the user but forms the solid platform for all subsequently implemented features. It does not demonstrate much user-facing functionality. Thus, the feedback we solicit from our tester should fall into two rough categories:

1. Did we demonstrate that we accomplished our developmental goal?
2. Are there any other fundamental capabilities that we need to think about before developing user-facing features in beta? ("What did we miss?")

3 Method

3.1 Approach

The focus of the UAT was on 'CRUD' or 'creation, retrieval, updating and deleting' observations. The observations demonstrate basic HTML form data entry: text fields, text areas, checkboxes, drop-downs, multi-select dropdowns, radio buttons and action buttons.

3.2 Testing Environment

The UAT was conducted at 11am on Tuesday 15 June in a virtual meeting by two test managers and one tester, each with their own equipment:

Tester: Nathan Reid (CSIRO)

- *Testing device:* Samsung Tab S6-Lite

Test Manager 1: Brian Ballsun-Stanton (Macquarie University, Technical Co-Director FAIMS Project)

Test Manager 2: Nuria Lorente (Macquarie University, Head of Software, AAO – Project Manager for the Australian Astronomical Optics FAIMS3 development team)

- *Testing Device:* iPad (native app)
- *Testing Device:* iPhone (native app)

3.3 Known Problems

The following **known problems** were declared at the beginning of the test:

- Syncing delay on iPad/iOS.
- Difficulty returning to records when using 'Save and new' button.
- Data loss (overwriting the version history) is possible if multiple devices update or delete the same observation while offline.

These known problems will be rectified during the next development phase.

3.4 Conduct of Tests

Under the guidance of the Test Managers the tester read through and followed the instructions in the **Alpha User Acceptance Testing Document** (Appendix A) to perform or observe set tasks, in sequence. They reported completion and discussion followed. Where relevant, comments were

entered in the Testing Document.

Tasks were categorised as 'Doable', 'Observable', or 'Attestable':

- **Doable tasks** are ones that the tester themselves should be able to perform on their devices or web-browser.
- **Observable tasks** are ones that the tester should be able to observe in consultation with the test-managers, as they usually involve interaction with database backends. These tasks are intended to demonstrate back-end functionality that currently lack frontend features for demonstration.
- **Attestable tasks** are ones which require test managers to state facts about the infrastructure and design of FAIMS3. This category has been included so that we can feature some of our design decisions and ensure that they fall within the expectations of testers, even though demonstrating them is either impossible or out of scope. Nevertheless, they represent a core FAIMS3 functionality, and are part of the UAT.

Following initial set up (including installing the app on selected devices), the tests followed three parts:

1. CouchDB and Project structure
2. Observation creation, retrieval, updating, and deletion on multiple operating systems
3. Dynamic Database interactions

Discussion (Part 4) followed.

See the UAT Testing Document for more details.

4 Results

4.1 CouchDB and Project structure (Part 1)

All tasks in **Part 1** were **successfully** performed or observed, attesting that:

- Module data schemas and UI schemas can imported from FAIMS2 modules to the FAIMS3 module creator. (Note: FAIMS2 logic files will not be translated to FAIMS3) (Task 1.1)
- FAIMS3 can assign Universal, offline, machine observation handles (using [UUID generation, version 4](#)) (Task 1.2)
- FAIMS3 supports constrained vocabularies (Task 1.3)
- FAIMS3 supports unicode and special characters in project level metadata (Task 1.4)
- FAIMS3 supports variable project metadata values (Task 1.5)

The key **Discussion** regarding the FAIMS3 database structure followed attestable Task 1.1 and asked:

Imagine you were a field director, preparing your project for FAIMS3. What would you expect the minimum “carry-over” from your FAIMS2 data and UI structures to be?

Tester’s response:

- Minimum: select location details, set SRID, keep SRID
- Critical: Being able to associate photos with site details
- Dropdowns, notes also necessary. Be able to tag locations.
- Very nice to have: QR stuff/printer.
- Never used bluetooth GPSes
- Getting-the-data-out part was very frustrating in FAIMS2. Simple and streamlined export needed for FAIMS3
- Need to be able to lock module creator stuff – so someone can’t stuff around while in the field taking data

4.2 Observation creation, retrieval, updating, and deletion on multiple operating systems (Part 2)

All tasks in **Part 2** were successfully performed or observed, attesting that:

- Observations can be created and saved in FAIMS3 (Task 2.1)
- Incomplete or Draft Observations can be saved locally to preserve a user’s work if they need to navigate away from their observation in progress (Task 2.2)

- FAIMS3 can take a GPS point by retrieving longitude and latitude from the device's internal location module (Task 2.3)
- Observations can be updated in FAIMS3 via CouchDB's robust append-only datastore (Task 2.4)
- Observations in FAIMS3 *can* be deleted as needed for app store compliance and also for storage reduction (Task 2.5)
- Observations can be created and updated in FAIMS3 while offline, and synchronised successfully once networking is restored (Task 2.6)
- FAIMS3 can load and run both using a web browser on iOS and as a 'native' application side-loaded onto the iPad/iPhone (Task 2.7)
- FAIMS3 supports the following data input elements: as text fields, text areas, checkboxes, drop-downs, multi-select dropdowns and radio buttons (Task 2.8)
- FAIMS3 supports unicode and special characters in the database (Task 2.9)
- Observations in FAIMS3 can be retrieved while offline (Task 2.10)

The tester raised a number of useful points about the interaction of data creation during **discussion**:

- Problems in hot weather, the GPSes on tablets would freeze, but not change nor notify. "Lying location services" (Task 2.1)
- One of the things, desired, is a prompt which 'hovers' to remind people to do something (create a duplicate) when some other record count was reached would be a desirable improvement. (Task 2.2)
- Make sure we can capture accuracy and elevation, and XYZ points. And we want to make sure that when the location data is poor-quality (old/hot tablets) we alert correctly. Make sure there is movement detection. (Task 2.3)

4.3 Dynamic Database interactions (Part 3)

All tasks in **Part 3** were successfully performed or observed, attesting that:

- UUIDs demonstrated in Part 2 are visible on the CouchDB interface (Task 3.1)
- Deleted observations can be undeleted in FAIMS3 (Task 3.2)
- Project definitions can be updated, dynamically (Task 3.3)

Tester's responses:

- UUID approach seems sensible
- minimum human identifier is "able to set letters" which are constants, and then map numbers and fields into the identifier
- Dumb auto-incrementing sufficient by default
- It would be good to be able to undelete that in the program

4.4 Discussion (Part 4)

The final part of the test provided an opportunity for the Tester to discuss whether they were satisfied that the goals for Alpha has been satisfied. The Tester responded:

I believe so. I am happy with everything that is working, and I can see where it's going for the next stage.

5 Evaluation

The FAIMS3 Alpha v0.1.0 UAT has demonstrated the goals for Alpha:

To demonstrate the foundational capabilities of FAIMS3. Specifically, loading a module from a specification, data entry on all OSes, and asynchronous data exchange on an append-only datastore.

have been fulfilled.

After four months of development, this is an excellent result for the FAIMS3 development team at AAO. They have reproduced, and significantly enhanced, the core functionality of FAIMS2 and established a good foundation for Beta development. The next phase of development will see the development of more user-facing features.

Lessons for Beta

While the Alpha UAT was focussed on foundational components of the FAIMS3 platform, the first interactions with users revealed some useful feedback. In addition to the overall endorsement of chosen design approaches (eg UUID generation), the interaction with and demonstration of features sparked a discussion of the need for human-readable Identifiers, camera integration and robust GPS point collection. These will be reviewed in the early stages of Beta development.

With regard to the format of the test, this was a productive method. Future UATs will require additional testers.

6 Appendices

6.1 Appendix 1

The following pages are a copy of the testing document produced inside the confluence wiki and exported to pdf.

Alpha User Acceptance Testing Document

The intention of this user acceptance testing is to explore, discuss, and confirm accomplishment of the goal of Alpha Development:

- To demonstrate the foundational capabilities of FAIMS3. Specifically, loading a module from a specification, data entry on all OSes, and asynchronous data exchange on an append-only datastore.

Where possible, this script will also be implemented by [@Rini Angreani](#)'s end-to-end testing such that videos are available demonstrating “observable” and “doable” sections.

From this development, FAIMS3 will be poised to develop for the full data-design to data-export workflow as well as for features required by researchers in the field. Without these eleven weeks of development, however, we would not have nearly as much assurance in our foundation. Therefore, the focus of this testing is on “CRUD” or “creation, retrieval, updating, and deleting” observations. The observations demonstrate basic HTML form data entry: text fields, text areas, checkboxes, dropdowns, multi-select dropdowns, radio buttons, and action buttons. As this data-entry is schematic, and designed to demonstrate our data integrity controls rather than field-appropriate data collection workflows, we do not have FAIMS2 specific features or our three target-modules implemented in this release.

Tester: Nathan Reid, CSIRO

- Testing device: Samsung Tab S6-Lite

Test Manager 1: Brian Ballsun-Stanton, Macquarie University, Technical Co-Director FAIMS Project

Test Manager 2: Nuria Lorente, Macquarie University, Head of Software, AAO – Project Manager for the Australian Astronomical Optics FAIMS3 development team.

- Testing Device: iPad (native app)
- Testing Device: iPhone (native app)

Table of Contents

- [Test Discussion](#)
 - [Task Categories](#)
 - [Feedback intentions](#)
 - [Caveats](#)
 - [Known Problems](#)
 - [Timeline](#)
- [Part 0 - Setup](#)
 - [Loading the app](#)
 - [Doable Task 0.1 - Native Android app](#)
 - [Doable Task 0.2 - Loading FAIMS3 in the browser with a clear cache.](#)
- [Part 1 - CouchDB and Project structure](#)
 - [Attestable Task 1.1 - Discussion of the faims 2->3 translator](#)
 - [Statements of fact:](#)
 - [Topics for discussion](#)
 - [Attestable Task 1.2 - Universal, offline, machine observation handles](#)
 - [Statements of fact](#)
 - [Topics for discussion](#)
 - [Observable task 1.3 - Constrained Vocabularies](#)
 - [Doable Task 1.4 - Unicode and special characters in project level metadata](#)
 - [Observable task 1.5 - Variable Project Metadata Values](#)
- [Part 2 - Observation creation, retrieval, updating, and deletion on multiple operating systems](#)
 - [Doable Task 2.2 - Incomplete/Draft Observations](#)
 - [Doable Task 2.3 - GPS and Taking a Point](#)
 - [Doable task 2.4 - Update Observations and the Draft Database](#)
 - [Task 2.4.1 - Updating an Observation](#)
 - [Task 2.4.2 - Draft database data preservation during updates](#)
 - [Doable task 2.5 - Delete Observations](#)
 - [How to send a tab offline in Chrome](#)
 - [Doable Task 2.6 - Offline data interactions](#)
 - [Observable Task 2.7 - iOS and iPadOS Support](#)

- Doable Task 2.8 - Data input elements
- Doable Task 2.9 - Unicode Support
- Doable task 2.10 - Retrieve (load) observations
- Part 3 - Dynamic Database interactions
 - Observable Task 3.1 - Observation UUID
 - Tester's response:
 - Doable Task 3.2 - Undeleting observations
 - Tester's response:
 - Doable Task 3.3 - Dynamic Observation Definitions
- Part 4 - Discussion

Test Discussion

Before we start the testing proper, here are some notes and asides about the structure of the test, our intentions, and known problems that have arisen as part of alpha development. We have judged that the known problems are not blockers for testing and will be resolved as part of our next phase of development.

Task Categories

Tasks in this user acceptance test can be categorised as “Doable”, “Observable”, or “Attestable”:

- **Doable tasks** are ones that the tester themselves should be able to perform on their devices or web-browser.
- **Observable tasks** are ones that the tester should be able to observe in consultation with the test-managers, as they usually involve interaction with database backends. These tasks are intended to demonstrate back-end functionality that currently lack frontend features for demonstration.
- **Attestable tasks** are ones which require test managers to state facts about the infrastructure and design of FAIMS3. This category has been included so that we can feature some of our design decisions and ensure that they fall within the expectations of testers, even though demonstrating them is either impossible or out of scope. Nevertheless, they represent a core FAIMS3 functionality, and are part of the UAT.

Feedback intentions

This Alpha prototype demonstrates core functionality of FAIMS 3's Electronic Field Notebook platform, which is largely invisible to the user but forms the solid platform for all subsequently implemented features. It does not demonstrate much user-facing functionality. Thus, the feedback we solicit from our tester and from user groups should fall into two rough categories:

1. Did we demonstrate that we accomplished our developmental goal?
2. Are there any other fundamental capabilities that we need to think about before developing user-facing features in beta? (“What did we miss?”)

Caveats



Feature in progress: A number of pages and features are marked by this symbol, denoting that the feature in question is in progress. Some such features (e.g. “Sign Up”) are not yet selectable, while others (e.g. “Recent Observations”) are intended to give the test user a flavour of our vision for the user interface going forwards.

Known Problems

- Syncing delay on iPadOS/iOS: There is sometimes a syncing delay, between the saved data on the device (i.e. PouchDB) and the CouchDB server. This appears to be related to (Apple's) user authentication on the device and is currently being investigated. If the iPhone/iPad is put to sleep and immediately reawakened, the data are correctly synced to the CouchDB server as expected.
- Clicking “Save and new” will clear the form and then launch a new observation form. If the back arrow is used (on browser or on android), it will take the user to the prior, but still cleared, form. This behavior will be refined as part of the project creation app development process in early beta.
- Data loss (overwriting the version history) is possible if multiple devices update or delete the same observation while offline. This is due to multi-user merge-conflict functionality not yet being implemented due to a dependence on the Authentication subsystem – which is, itself, not yet implemented.

Timeline

- AAO: Feature-freeze: 28 May 2021
- Alpha Release: 11 June 2021
- AAO: Alpha prototype for sideloading to CSIRO's Nathan Reid: 14 June 2021
 - @ Brian Ballsun-Stanton, @ Nathan Reid, and @ Nuria Lorente to conduct tests on 15 June in the morning.
- Publication of UAT results and public announcement of Alpha, 21 June 2021
- FAIMS3: This script, Nathan's feedback, and tools to run UAT will be released to TAG for additional feedback to guide late beta development.

Glossary: [Glossary](#)

Part 0 - Setup

Loading the app

Doable Task 0.1 - Native Android app

Relates to <https://faimsproject.atlassian.net/browse/FAIMS3-174>

We plan to put FAIMS3 on Google Play and the Apple App Store during a working, public, beta. At present, side-loading the app onto mobile devices is required. We do not plan to support mobile browsers with FAIMS3 due to constraints that operating system manufacturers place on javascript engines in mobile browsers. Instead, we plan that visiting FAIMS3 with a mobile browser will result in a redirection to the appropriate app-store and then loading that specific page of FAIMS3 in the mobile app.

1. If repeating the test, consult with test managers on the best approach to clear cached app data.
2. On your android, go to <https://github.com/FAIMS/FAIMS3/releases/tag/v0.1.0-alpha>
3. Click assets, then choose `app-debug.apk`. Download and install. Work with test managers to resolve device-specific installation requirements.

Tester's response:

- No response

Doable Task 0.2 - Loading FAIMS3 in the browser with a clear cache.

We anticipate that testers may wish to conduct this test more than once. This step is to ensure that all user acceptance testing starts with a clean slate.

1. Visit <https://alpha.3.faims.edu.au/> from chrome on your computer.
2. Press F12 on your keyboard
3. In the sidebar which appears, choose the menu item `Application`
4. In that sidebar's sidebar, choose either `storage` or `Clear storage`
5. Press `Clear site data`
6. Close the inspection sidebar
7. press and hold shift, then refresh the page, then release shift.
8. Quit your webbrowser and re-open it
9. Inform test managers to reset the testing database. Wait until they inform you that testing can proceed.

Tester's response:

- No response

Part 1 - CouchDB and Project structure

In FAIMS3, we are calling an entire data collection undertaking a "project" In FAIMS2.6, this was termed a "module." Projects, in the future, will be able to support multiple observations (tabgroups), multiple data-collection pages per observation (tabs), and child observations and cross-references to other observations (relationships).

Attestable Task 1.1 - Discussion of the faims 2->3 translator

During Alpha development, FAIMS3 focused only on the core infrastructure of observation creation and sync. As such, when needing to touch on our three target projects (Oral History, CSIRO Geochemistry, and Lake Mungo), we worked on a program which could translate project-specification code from prior versions of FAIMS to the current version. The current state of the translator is not ready for demonstration, because FAIMS3 lacks multiple-tab and multiple-observation support. However, we have demonstrated that we can parse old data and ui schemas, and plan to use this as one of the inputs into the web-project generator.

Statements of fact:

- Module data schemas and UI schemas imported from FAIMS2 modules will be available for the FAIMS 3 module creator.
- FAIMS3 Alpha can load some of these converted modules. They have not been tested, and since Alpha does not support multiple observation-types nor multiple tabs in a page, this functionality was not implemented in this acceptance test.

Topics for discussion

- Imagine you were a field director, preparing your project for FAIMS3. What would you expect the minimum “carry-over” from your FAIMS2 data and UI structures to be?
 - **Tester’s response: Minimum, select location details, set SRID, keep SRID**
 - **Critical: Being able to associate photos with site details**
 - **Dropdowns, notes also necessary. Be able to tag locations.**
 - **Very nice to have: QR stuff/printer.**
 - **Never used bluetooth GPSes**
 - **Getting-the-data-out part was very frustrating in FAIMS2. Simple and streamlined export needed for FAIMS3**
 - **Need to be able to lock module creator stuff – so someone can’t stuff around while in the field taking data**
- We do not plan to translate the *logic files* of FAIMS2 as we do not have feature parity and language transpilation is an extremely labour intensive task. We think that presenting users with defined observations, tabs, field-names, descriptions, and controlled vocabularies will be sufficient for effective conversion. Do you think this is the correct judgement call?
 - **Tester’s response: yes.**

Tester’s response:

Attestable Task 1.2 - Universal, offline, machine observation handles

Relating to: <https://faimsproject.atlassian.net/browse/FAIMS3-188>

One challenge in FAIMS2 was that users were required to login as part of observation-creation needs. A observation’s UUID (the internal handle that the app used to track the observation) was created by conflating the local user ID with a millisecond timestamp. The led to the chance that two devices with the same logged-in user would create the same observation handle if two people clicked save in the same millisecond. FAIMS3, on the other hand, uses **UUID-4** which is a widely-used mechanism for having globally unique identifiers without reference to a central system. The odds of a observation handle collision are so small per Project, that we would need 103 trillion observations to have a 1-in-a-billion chance of conflict, and are therefore sufficiently reliable to support scientific observation. Observation handle conflicts will be discussed in the third (Production) phase of the project. However, as we estimate a maximum of 300,000 new observations created per project, we do not think the odds of this happening are worthy of concern.

Statements of fact

- We are using **UUID generation, version 4**
- The risk of collision is significantly less than that of FAIMS2.6 which only required two people logged in as the same user to create a observation in the same millisecond.
- Detecting observation handle duplicates is part of our backlog, as an additional testing item for merge-conflict detection, which is scheduled for beta development.
- We accept the risk of one in a billion when a project has a trillion observations, noting that projects of that scale will require custom development in any case.

Topics for discussion

- Do you think the odds of collision (one-in-a-billion if an individual project has 103 trillion observations) are sufficient for the needs of the project?
 - **“I can live with those odds”**
- Do you think this risk is a better approach than the user+time approach of FAIMS2?
 - **“Never used the UUID in faims2”**

Tester’s response:

Observable task 1.3 - Constrained Vocabularies

Relating to: <https://faimsproject.atlassian.net/browse/FAIMS3-66>

One of the fundamental features of FAIMS2.6 was the ability to support nuanced constrained vocabularies: ones that could handle hierarchies of vocabulary, ones that could support “Picture dictionaries” and ones that could reference linked open data vocabularies. During Alpha, our goal was to prevent our design decisions from precluding these development objectives – even though actual development of these features was out of scope during Alpha development.

- i Of note, we do not currently support “other” field-display as part of a constrained-vocabulary choice, but our design does not preclude this feature. The “other” support in the issue above has been migrated to a early-beta feature specification.

This task requires a test manager to demonstrate that our data specifications can accept constrained vocabularies with more than one level:

1. Open the CouchDB database: https://alpha.db.faims.edu.au/_utils/#
2. Click `metadata-test_proj-<UUID>`
3. Click `ui-specification`
4. Navigate to line ~318
5. Demonstrate how we handle subsidiary key-value pairs for constrained vocabularies
6. Discuss how this key-value design will also be able to handle metadata like picture urls, descriptions, certainty, and annotations.

Tester’s response:

- “Yes, it is good, necessary for all of them”

Doable Task 1.4 - Unicode and special characters in project level metadata

Relating to: <https://faimsproject.atlassian.net/browse/FAIMS3-167>

Relating to: <https://faimsproject.atlassian.net/browse/FAIMS3-172>

We want to make sure we have unicode character support for international projects baked in from the start. This task demonstrates that we preserve right to left character encoding and support those unicode languages.

1. Navigate to: <https://alpha.3.faims.edu.au/> in your web browser
2. Navigate to the projects view by selecting the `View all Projects` link in the home page (in the “MY PROJECTS” panel) or choose, from the hamburger menu () based sidebar, `Projects/Test Project “Astrosky”`.
3. Note the presence of characters like `(,) , “` and `-` in module names.
 - a. Motivation: in FAIMS 2.6, special characters like those above could cause module metadata to break
4. Note that one project lead is named `Robert'`; `DROP TABLE Students;--`
5. Copy the lead institution (`Lead Institution: -`) (by dragging from left to right) from the web-app into the space provided:
 - a. Paste here: `Lead Institution: -`
 - b. Place your cursor into the word `institution`, and tap the right arrow key. Note that the cursor behaviour changes to “Right-to-left” when entering the Hebrew text.

Tester’s response:

- “Fantastic”

Observable task 1.5 - Variable Project Metadata Values

Relating to: <https://faimsproject.atlassian.net/browse/FAIMS3-167>

In Alpha, we have hardcoded “Project lead” and “Lead institution” project metadata, to demonstrate that we can store and retrieve arbitrary strings in project level metadata. We plan, in our backlog, to support arbitrary project metadata which will be able to store pdfs and static files as well as strings. This will allow project leads to store manuals and reference material at the project level.

1. Navigate to: <https://alpha.3.faims.edu.au/> in your web browser
2. Click projects
3. Observe the name of [project lead](#).

4. Ask that the test manager access the database at: https://alpha.db.faims.edu.au/_utils/#login
5. They should then load `metadata-test_proj-<UUID>` and then click on `project-metadata-project_lead`
6. The tester should specify what name to adjust the project lead to.
7. Navigate back to FAIMS3 Projects and refresh the page.
8. Observe the change of the name of `Project lead`

Tester's response:

- "It does what it says on the tin"

Part 2 - Observation creation, retrieval, updating, and deletion on multiple operating systems

Relating to

- <https://faimsproject.atlassian.net/browse/FAIMS3-173?focusedCommentId=11192&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-11192>
- <https://faimsproject.atlassian.net/browse/FAIMS3-187>

The foundation of FAIMS3 is offline-capable multi-platform synchronization on user-defined project specifications. This objective was the core objective of alpha development.

Doable Task 2.1 - Observation creation

Relates to: <https://faimsproject.atlassian.net/browse/FAIMS3-25>

Fundamental to field data creation is the act of creating observations. In this task, we will explore the creation and synchronisation of observations. There is a great deal of nuance in the practice of observation creation. During Alpha, the priority was to demonstrate robust data integrity rather than expanding the feature set into multiple tabs per observation and multiple observations per project. These required features for field data collection are planned for early beta. Here, in this user acceptance testing, we are demonstrating the bald fact of observation creation and synchronisation across multiple operating systems and devices.

Also, we decided to emulate a "save-and-new" data collection workflow. Prior to test finalisation, the button also could be set to "finalise the observation" (without leaving it, allowing for future updates but saving it to the central database ready for synchronisation) as well as save-and-exit which behaves much like the `UPDATE` button does in task 2.4. We anticipate that field directors or their project designers will be able to control this functionality on an observation by observation basis – to suit their particular workflows.

The data collection elements on this page were chosen solely to demonstrate the breadth of currently implemented data collection capabilities (Task 2.8) and not as part of any project workflow. While we did implement the underpinnings of translation capabilities for our three target projects (Task 1.1) – we are missing the necessary user experience elements to make those projects persuasive test instruments. This development is planned for early beta.

i The `SAVE AND NEW` button, in alpha, demonstrates that we have client side data-integrity checking. It also demonstrates that we are taking care not to "pollute" the central database with invalid observations. Only after the user decides a observation can be synced and after it passes data-integrity checks (as specified by the field director as project designer) will it be written to the central database. For some projects, this will be immediately – but for e.g. citizen science projects, data quality standards can be enforced so that as someone is walking along a beach and collecting data, no action is performed against the central server until the observation is complete. This "Draft mode" allows a user to exit the observation or app and resume data entry (i.e. needing to have a observation per beach that encompasses multiple variables along the walk) without needing to write partial observations to the central datastore.

1. Browser and android device should load FAIMS3, and navigate to the projects page,
2. All devices should click on `Test Project "Astrosky"`
 - a. Click `+ NEW OBSERVATION` (if on a narrow-screen mobile device, by accessing the kebab menu on the top right of the projects pane), enter a sample email address into the form:
 - i. `android-<testername>@faims.edu.au` on the android
 - ii. `chrome-<testername>@faims.edu.au` on the browser
3. Fill out all other fields in the form, until the red error box above `Save and new` goes away.
4. Click the blue `SAVE AND NEW` button at the bottom of the form
5. Observe that, after clicking `save and new`, a cleared observation form is presented to the user for ease of data collection

i There is no requirement to return to the Project level and select the `+ NEW OBSERVATION` button after each observation save). This functionality (stay on current observation, clear and go to new observation, or return to the calling page) will be specifiable by the project designer once we implement that functionality in beta.

1. Inspect the top-right header showing “ Observation successfully created” in a green box.
2. With the new, blank, observation, continue to task 2.2

Tester’s response:

- Problems in hot weather, the GPSes on tablets would freeze, but not change nor notify. “Lying location services”
- “This is pretty straightforward”
- Needs to highlight where the form sees an error – yes, we need more.

Doable Task 2.2 - Incomplete/Draft Observations

Relating to: <https://faimsproject.atlassian.net/browse/FAIMS3-25>

In FAIMS3, one of our goals was to ensure higher data quality throughout a project’s lifecycle and to allow users to resume work on an observation without reducing the data quality of the database and littering other users’ views with temporary observations. As such, AAO has implemented a “Draft database” feature. Any observation that is either incomplete or that fails basic data-integrity checks (fields required for an observation to be valid, or where only numbers or allowed, or other validation) will be saved locally, to preserve a user’s work if they need to navigate away from their observation in progress – without spoiling the database.

1. In Test Project “Astrosky” click + New Observation on both chrome and android.
2. Enter a sample email address into the form:
 - a. draft-observation-android-testername@faims.edu.au on the android
 - b. draft-observation-chrome-testername@faims.edu.au on the browser
 - c. Wait 2 seconds but **do not** click SAVE AND NEW
 - i. Note the blue box at the top of the form notifying
“ Draft last saved a few seconds ago DD Mmm dd hh:mm:ss GMT+1000 (AEST)”
 - d. Leave the observation entry form
 - e. Load a different observation and then return to the projects page
 - f. Click + New Observation
 - g. **Did your draft observation reappear?**
 - h. Wait 10 seconds for any sync to complete
 - i. Quit the app (close the app tab in your browser, or on mobile device, ensure that the app has been quit through the app switcher)
3. Have the test manager confirm that the partial observation does not exist in the CouchDB database.
4. Restart the app, and return to the projects page
5. Observe the same number of observations as previously. Inspect observations if desired.
6. Load your “draft” observation by clicking the + NEW OBSERVATION button – it should pre-populate with your previously entered data.
7. Ensure that your data are still present, despite not being synced to the server, on both devices, and the neither device shows the other device’s draft observation.
8. Finish the observation, click save and new, return to the projects page, and observe that saved observations that were previously in draft are now synced to all devices.

Tester’s response:

- One of the things, desired, is a prompt which “hovers” to remind people to do something (create a duplicate) when some other record count was reached would be a desirable improvement.
- All Worked well

Doable Task 2.3 - GPS and Taking a Point

Relating to <https://faimsproject.atlassian.net/browse/FAIMS3-170>

During Alpha, we wanted to demonstrate basic functionality for interacting with easily supported device sensors. As such, we can get longitude and latitude from the device’s internal location module. This component also demonstrates that we can run logic when pushing buttons – necessary capabilities for any interactivity in the future.

- i** Note, that as this default location functionality exists as part of standard functionality, it pulls capabilities that may be contrary to scientific objectives. Specifically, *the device* will return a cached location when it doesn’t have positive sky view rather than providing an error. Improving geolocation capabilities with internal and external sensors is in our planned backlog.

1. Go to the `index/Home` page, and click `add` in the add new observation box. (This is the same as clicking new observation, but shows a different page).
2. Ensure that your device's GPS is enabled and that you have good sky view.
 - a. On a mobile device, ensure that you have appropriate permissions granted to the app
3. Start a new observation
4. Press the `Take Point Button`.
5. Observe your longitude and latitude output rendered beside the button, and in the developer side-panel as a JSON object
6. Press the `Action` button, note that the JSON debug view now shows `Change!` rather than `hello`
7. Finish the observation
8. Click `save and new`
9. return to the projects page
10. Load the just-created observation
11. Ensure that `location` and `change` are still present in the data

Tester's response:

- Make sure we can capture accuracy and elevation, and XYZ points. And we want to make sure that when the location data is poor-quality (old/hot tablets) we alert correctly. Make sure there is movement detection

Doable task 2.4 - Update Observations and the Draft Database

CouchDB provides an append-only datastore, where revisions to observations are always stored as new observations as part of its MVCC replication strategy. This provides a robust foundation for our alpha prototype, though as some sources note (1, 2, 3, 4) relying on this alone may cause problems when we are detecting and dealing with merge conflicts, we still are able to demonstrate append-only documents showing the possibility of looking at earlier versions.

In this task, we demonstrate the robust append-only datastore provided by CouchDB. We also demonstrate how leaving a new observation-in-progress to update a different observation, or editing multiple observations in pseudo-parallel (i.e. not finalising the updates to be synced to other devices) is entirely supported by the draft-database infrastructure provided.

Task 2.4.1 - Updating an Observation

Relates to:

- <https://faimsproject.atlassian.net/browse/FAIMS3-190>

1. Navigate to the projects view
2. Load a observation of your choice.
 - a. Note the first 6 characters of the UUID here (for ease of future reference) **459d03**
3. Edit data. Wait for the draft to save, leave the observation
4. Load the observation on a different device – observe that your changes have not propagated
5. On the original device, start a new observation, wait for the draft to save, leave the observation
6. Still on the original device, return to the observation you were initially editing, note that your edits were preserved
7. Finish editing data, press `UPDATE`
8. Edit data again, press `UPDATE` again
9. Click new observation and ensure that your draft new observation is still there.
10. Return to the edited observation.
11. Verify that a new revision has been created by selecting the `REVISIONS` tab at the top of the screen. This will show the ID for each of the revisions of the observation.
12. Wait for the app to sync
13. Ask the test manager to demonstrate the observation revision in CouchDB's concurrency view by having them copy the revision ID from the revisions list and showing the JSON of the original observation. (getting the json of the observation and appending `?rev=<REV-ID>` will show the prior observation.

✘ Many sources on the internet caution against using the CouchDB version history as an append-only datastore. as the version history is overwritten on offline conflicting updates to the same observation. However, implementation of a version-control style audit trail requires user authentication to be implemented and is therefore planned as development for early beta. As merge-detection and conflict resolution was not part of the alpha spec, this issue was not detected during alpha development. CouchDB and PouchDB have [elegant conflict detection and handling mechanics](#) such that once user authentication is implemented, conflict detection and resolution will be effectively handled.

Tester's response:

- "for what's there, it works well. Other things as they come"

Task 2.4.2 - Draft database data preservation during updates

This task demonstrates that the "Draft database" also caches edits-in-progress as well as new-observations-in-progress. Here we are performing the inverse of task 2.2 – editing a observation, pausing our edits, creating a new observation, and then resuming our edits.

1. Edit an observation on the browser but do not press update
 - a. What are the first 6 characters of this observation's UUID?
2. Return to the observations list
3. On the browser, Press + New Observation
4. Fill out a new observation, allow autosave to occur, press SAVE AND NEW
5. Leave the New Observation view
6. On the browser, Resume editing the observation in step 1
7. Observe persistence of changes
8. Load the same observation in a different platform
9. Observe that the draft database has not synchronised the latest, unsaved changes, but *has* synchronised the new observation. Leave the observation.
10. In the first browser, press update on the edit-in-progress
11. Allow the application to sync
12. Observe observations on other devices after they sync – verify that only the latest data appears, but the list of all revisions is shown in the REVISIONS tab.
13. Ask the test manager to demonstrate revision history in CouchDB, compare revision history to data created in prior steps

Tester's response:

- Important to demonstrate camera attachments in FAIMS3 early beta
- "Cool"

Doable task 2.5 - Delete Observations

Relates to:

- <https://faimsproject.atlassian.net/browse/FAIMS3-29>
- <https://faimsproject.atlassian.net/browse/FAIMS3-179>

It is important that researchers be able to see and work with valid records. Therefore, records need to be "deletable" such that they are not mistaken for valid data. However, one of the central tenets of FAIMS is to "never lose data" – as such, observations are not expunged on deletion, but rather, hidden. We plan to provide undeletion capabilities in beta as well as a way of expunging deleted observations for duly authorised field directors deep in a control-menu somewhere – mostly as a mechanism for complying with moderation rules in the various app stores and as a way of reducing storage consumption in case FAIMS is deployed to low-storage environments. However, no-risk deletion *qua* hiding is central to maintaining a clean project.

1. Load project Test Project "Astrosky" by going to the hamburger menu () and clicking the project name.
2. While online, all devices should create a observation with "deleteme-testername@[Device.OS]" (deleteme@app.android, deleteme@chrome.linux, deleteme@app.iOS, etc..) in the email address field. Click SAVE AND NEW and allow sync to occur by waiting a few moments.
 - a. Note the first 6 characters of each UUID of the created observations here for future reference:
 - i. Chrome:
 - ii. Android: **459d03**
 - iii. iOS:
3. All devices should return to the Test Project "Astrosky" project page and should confirm visibility of all created observations.
4. The android device should delete its observation by opening the observation, and choosing DELETE OBSERVATION in the META tab of the blue panel at the top of the screen. A dialogue will appear asking you to confirm this. Confirm deletion.
5. Observe that the observation has disappeared from the list of RECENT OBSERVATIONS.
6. Observe the absence of the observation on other devices (so long as a sync has occurred, so a few moments' pause may be required)
7. Android should toggle sync to off. *N. B. this will pause sync only while this page is loaded.*
8. Chrome to delete its "deleteme". Observation should still be visible on android, but not on iOS.

9. Android sync enabled
10. Observation should no longer be visible on the other two devices
11. Android and Chrome sync disabled
12. iOS to edit its observation, save, then delete its observation and allow sync to occur
13. all sync enabled, confirm iOS observation is hidden
14. Test manager to demonstrate that deleted observations still live on the CouchDB database, including any edits made

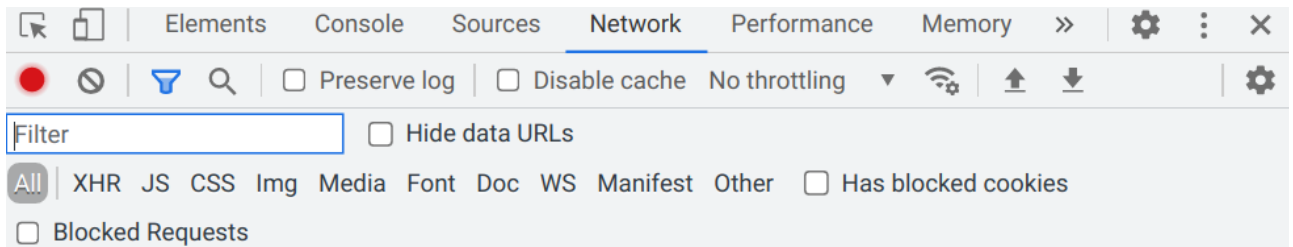
Tester’s response:

“Cool. I think that all works perfectly well.”

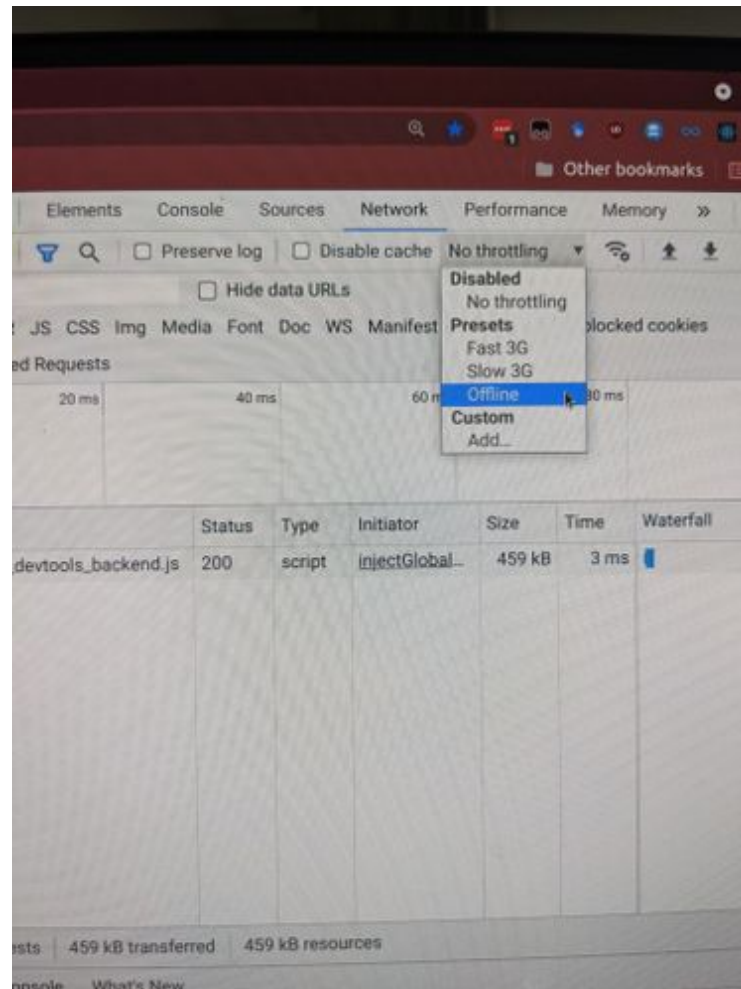
How to send a tab offline in Chrome

The following tasks will be demonstrating with offline mode. As FAIMS3 is a “Single page application” offline mode can be accomplished in the browser without taking the computer offline. To take a browser page offline, follow the following steps:

- Press F12 or shift-ctrl-i for the developer tools console
- In the sidebar or window that appears, at the top of the sidebar, there will be many tab selectors:



- Click the one that says network
- Choose the dropdown that currently says “no throttling”
- Choose offline.
-



- To reenabling networking, choose no throttling

⚠ The sync toggle applies only to the current page, and does not persist across page loads. While entirely functional, refreshing the page or going to different pages will produce results contrary to intuition. Setting the page offline is to be preferred for any tasks that require page navigation.

Doable Task 2.6 - Offline data interactions

Relating to: <https://faimsproject.atlassian.net/browse/FAIMS3-181>


Offline capability is central to FAIMS3's design objectives. We aim to provide systems for robust multi-device offline data creation, retrieval, editing, and deletion. A significant amount of work during alpha was aimed towards designing and proving this capability.

1. The tester should **turn off networking** (See How to set a tab offline in Chrome above) to the FAIMS3 browser tab (as defined above) and disable wifi and mobile data on the device (but do not engage aeroplane mode, to preserve GPS connectivity.) The iOS device should turn on aeroplane mode.
2. Load module "Test Project "Astrosky""
3. Perform the **Draft Database** and **Observation Creation** tasks (2.1, 2.2), but without syncing
4. Press take point, and see the tester's Longitude and Latitude display, after satisfying any browser or OS permission requirements.

i Note that if take point is pressed in aeroplane mode it will return `location unavailable`, because there is positive knowledge of no location. This is distinct from using cached location without skyview (i.e. when in a tunnel.)

1. Save and validate all created observations with `SAVE AND NEW`
2. For the browser:
 - a. Turn protocols on and allow the device to sync (this should only take a few seconds).
 - b. Observe that no newly created observations appear


3. On Android,
 - a. Turn wifi on
 - b. Observe that the browser's observation appears in the RECENT OBSERVATIONS panel
4. On the browser
 - a. Observe that the Android's observation appears in the browser
5. Ask that the test manager performs the same steps on the Apple device(s).
6. The test user should now disable networking within the chrome FAIMS3 tab, Android should turn on aeroplane mode (which includes GPS), iOS should turn on aeroplane mode.
7. The browser and Apple device should edit their "Textarea field label" field to say <device> offline edit.
8. On android, navigate to a previously created observation,
 - a. **Note the observation UUID here:**
 - b. click on meta, then delete observation, then delete
9. While offline, all devices can view, edit, and delete observations created on other devices.

 Alpha does not support merge conflict detection. Therefore, other-device observation editing and deletion is **not** part of the formal alpha UAT as it may create inconsistent results. While very careful experimentation is possible with offline/multi-device editing, sync between each edit to avoid confusing outcomes and hidden error states with record conflicts.

Tester's response:

- Yes, content – noting iPad weirdness.
- Test 2, starting from data off while app is completely off
 - Looks like it's hunky dory.

Observable Task 2.7 - iOS and iPadOS Support

 Due to a known problem in the alpha build, the test managers will be demonstrating iPad and iPhone support.

One significant criticism of FAIMS2 was that the app was android-only. By extending FAIMS3 to the browser and to native iPadOS and iOS support, we hope to dramatically extend our possible userbase.

Relates to

- <https://faimsproject.atlassian.net/browse/FAIMS3-174>
 - <https://faimsproject.atlassian.net/browse/FAIMS3-175>
1. Observe that iOS can load and run FAIMS 3 both using a web browser on iOS and as a "native" application sideloaded onto the iPad / iPhone.
 2. Satisfy yourself that the capabilities specified in prior tasks adequately demonstrate iOS and iPadOS support.

Tester's response:

- "I hate apple"

Doable Task 2.8 - Data input elements

Relates to:

- <https://faimsproject.atlassian.net/browse/FAIMS3-66>
- <https://faimsproject.atlassian.net/browse/FAIMS3-73>

While we do not have many user-facing features compared to the mature FAIMS2 product, we wanted to demonstrate that we could interact with HTML form elements in a systematic fashion. Therefore, part of the spec of alpha was to demonstrate input fields, text areas, dropdowns (but not hierarchical dropdowns nor picture galleries), checkboxes, radio buttons, and action buttons. We also wanted to demonstrate that we can show a number pad for certain text input areas. Of note, of the tasks above, we downscoped FAIMS-66 so that extending a dropdown with an "other" textbox was out of scope for alpha. It is planned to be implemented for when we support field level metadata (annotation, certainty, and descriptions.)

1. Load project Test Project "Astrosky"
2. Verify that the dropdowns work
3. Verify there can be multiple entries checked in in the checkbox vocabulary

4. Make sure that radio buttons:
 - a. Can have one and only one selected at a time
 - b. Cannot be unselected (because that's part of the original rationale of radio buttons)
5. Observe the constraints available on text entry fields:
 - a. Some fields (text area) should accept any input
 - b. The 0-20 Integer field should be a number only. This should accept zero and decimal whole numbers, and reject negative and non-integers
 - c. Other fields should have basic data integrity checks applied

Tester's response:

- Annotations were something used on rare occasion. If it wasn't there as a field they could see, people just didn't use it. The only time there was a caveat, going back in and manually entering a GPS coordinate
- "I'm happy with that"

Doable Task 2.9 - Unicode Support

Relating to <https://faimsproject.atlassian.net/browse/FAIMS3-168>

1. Edit data in saved observation on the browser
 - a. Copy the following (or similar phrase from <https://r12a.github.io/scripts/arabic/>)
 - i. .
 - ii. Make sure you copy the full line by triple-clicking on the line (this will capture the invisible rtl character).
 - iii. Paste into the text field. Ensure that the cursor is moving in the appropriate (rtl) direction when using the arrow keys
 - b. Copy the following into the Favourite Colour or Textarea field:
 - i. (Each of these are more than two-byte emoji sequences from the latest standard, so will most likely break things if there are things to be broken this way)
 - ii. Or go to <https://unicode.org/Public/emoji/13.1/emoji-test.txt> (Copy symbols you think might be used in field research or you think the app might have trouble rendering).
 - c. Copy each of the following lines into a field – note that confluence does not render some of the characters correctly (or any others from <https://github.com/minimaxir/big-list-of-naughty-strings/blob/master/blns.txt>):

```
i.
AĖÉÛßªñ

o invokè the hive-mind represeting chaos.
<script>alert(0)</script>
ABC<div style="x\x3Aexpression(javascript:alert(34))">DEF
,./;'\[]\-= <>?:"{}|_+ !@#$$%^&*() `~
Power h
```

2. Press SAVE AND NEW as desired for above tests.
3. Double check observation appearance on all devices and in the DB

Tester's response:

- Everything works and looks as it should

Doable task 2.10 - Retrieve (load) observations

Relates to:

- <https://faimsproject.atlassian.net/browse/FAIMS3-190>
- <https://faimsproject.atlassian.net/browse/FAIMS3-180>

All of the prior tasks involved loading records, though no one task focused on it. Please use this task to satisfy yourself of the reliability of record loading on and off-line.

1. All devices turn wifi on/airplane mode off.
2. Observations appear on all devices with edits and deletes applied.
3. Take each device offline in turn and ensure that records are still present.
4. Quit the app, and – while offline, launch the app (or page) If you set a browser page to offline, you'll need to "visit" <https://alpha.3.faims.edu.au/> in that browser page (rather than any child page to have the cached single-page application load.
5. All non-deleted observations are visible on all devices after sync, and can be loaded.

Tester's response:

- It was really good

Part 3 - Dynamic Database interactions

Observable Task 3.1 - Observation UUID

Relating to <https://faimsproject.atlassian.net/browse/FAIMS3-188>

1. After the active task "Observation creation and loading" is completed, [@ Brian Ballsun-Stanton](#) to demonstrate on the CouchDB interface that all observations created during part 2 have unique machine-readable identifiers.

Tester's response:

- It seems sensible
- minimum human identifier is "able to set letters" which are constants, and then map numbers and fields into the identifier
- Dumb auto-incrementing sufficient by default

Doable Task 3.2 - Undeleting observations

Relating to: <https://faimsproject.atlassian.net/browse/FAIMS3-29>

Related to: <https://faimsproject.atlassian.net/browse/FAIMS3-178?focusedCommentId=11670&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-11670>

1. After the active task, "Delete Observations" is completed, demonstrate that deleted observations are still present in CouchDB with a "deleted" flag set.
2. After performing tasks from part 2, observe the CouchDB observation utility and observe that all observations are present
3. Test manager to undelete a observation by setting `deleted` to false
4. Observe observation's presence on device
5. Load the observation and make edits to it

Tester's response:

- It would be good to be able to do that in the program

Doable Task 3.3 - Dynamic Observation Definitions

<https://faimsproject.atlassian.net/browse/FAIMS3-171>

1. Make a new observation while in the browser. Observe fields. Enter data into the fields, save
2. Ask the test manager to edit the project definition while in CouchDB. Add a text field with label desired by the user. The text field template is below, **<replace me>** should be replaced with the desired label. This should be inserted just before multi-str-field. Then, inside "fvIEWS"."start-view"."fields", add "new-user-field".

⚠ There are 2 potential cases where entering the wrong thing into the db causes issues:

- if you duplicate a field in a view, the field keeps appearing, causing the page to lengthen;
- if you do not choose a new name and id in the component-parameters, then the fields get tied together, and the form library appears to get confused.

In short: **Don't copy-paste fields in couchDB.**

This is only a problem due to manual editing of the database. Our major deliverable in beta, the module-creation app, will protect users by limiting direct access of the database. Part of that app will provide for dynamic manipulation of modules.

1. Wait for the local (i.e. testing device) FAIMS3 to sync with the CouchDB
2. Make a new observation, note presence of new field. Add data to that field (and others) and save.
3. Load original observation. Verify that the new text field appears, empty (because it didn't exist in the original observation), or containing the field initial value if one was specified.

Test Manager JSON:

```

"new-user-field": {
  "component-namespace": "formik-material-ui",
  "component-name": "TextField",
  "type-returned": "faims-core::String",
  "component-parameters": {
    "fullWidth": true,
    "name": "new-user-field",
    "id": "new-user-field",
    "helperText": "<replace me>",
    "variant": "outlined",
    "InputProps": {
      "type": "text"
    },
    "SelectProps": {},
    "InputLabelProps": {
      "label": "<replace me>"
    },
    "FormHelperTextProps": {}
  },
  "validationSchema": [
    [
      "yup.string"
    ]
  ],
  "initialValue": "Hello I'm new"
},

```

ⓘ N.B. The Test Manager should make sure to add the `new-user-field` to the field list at line 520 or it won't show up.

Tester's response:

"I like it"

Part 4 - Discussion

Test managers and tester should now engage in unstructured exploration of FAIMS3 Alpha, until tester is satisfied that the expensive sentence for Alpha:

“To demonstrate the foundational capabilities of FAIMS3. Specifically, loading a module from a specification, data entry on all OSes, and asynchronous data exchange on an append-only datastore.”

has been fulfilled.

Tester’s response:

“I believe so” “I am happy with everything that is working, and I can see where it’s going for the next stage.”