Moving FABS into BFYS

Overview

The Get Help Buying for Schools (GHBS) service comprises two Rails applications `find-a-buying-solution` (FABS, now referred to as the GHBS website) and `buy-for-your-school` (BFYS), which includes Case Management System (CMS), Energy for Schools and School Collaboration.

FABS is hosted on Heroku and BFYS is hosted on Azure.

Azure is DfE’s preferred hosting provider, so we need to move FABS from Heroku to Azure.

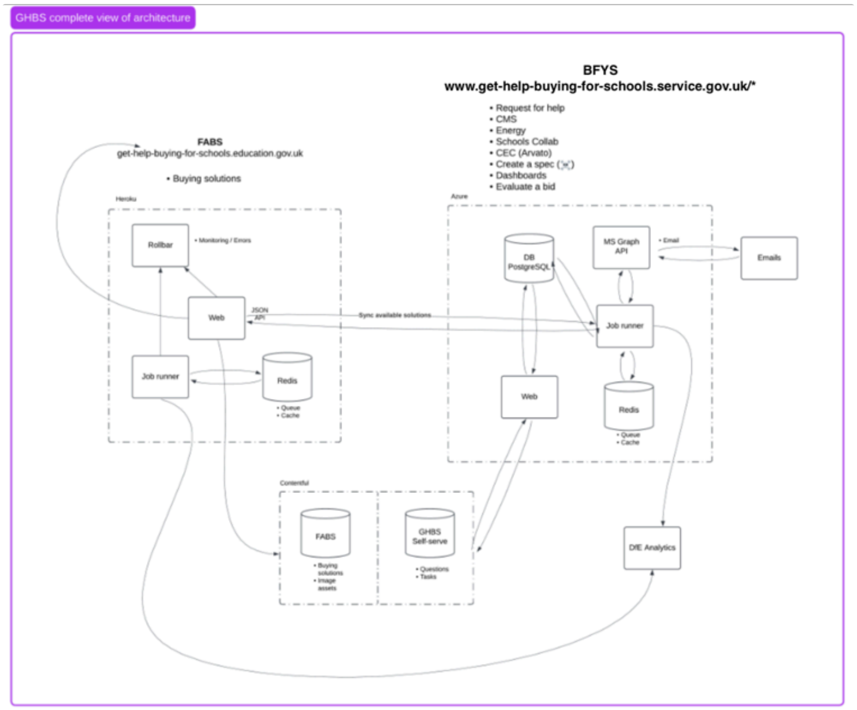
The best way to do this would be to combine the two applications by merging the code from the FABS repo into the BFYS repo. This provides several benefits:

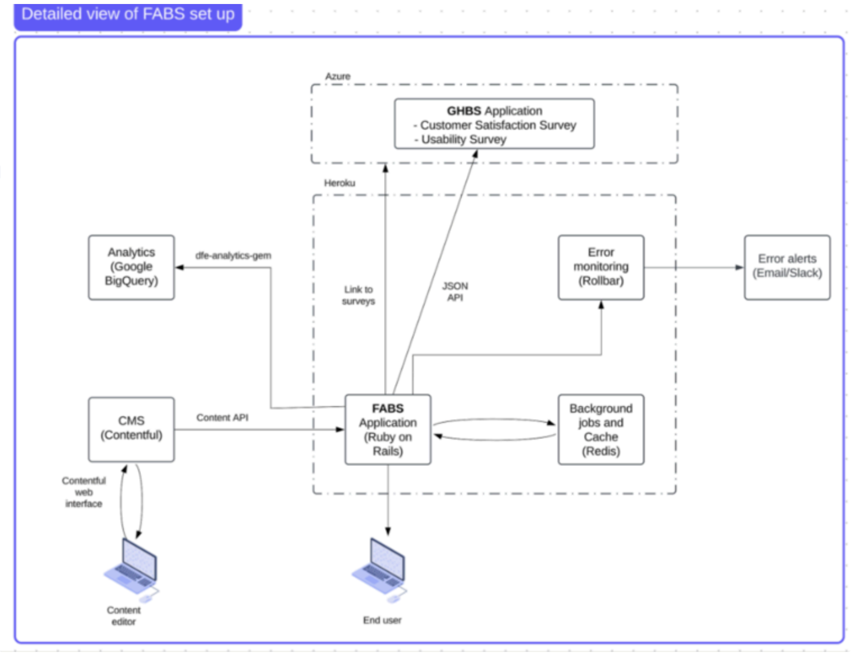
- Reduce development and infrastructure costs
  - No need for new Azure subscription and hosting resources
  - Reduced usage of Heroku
- Remove data synchronisation overhead between FABS and BFYS
- Leverage existing cookie/DPIA compliance from CMS
- Use a single Contentful space to enable rapid content updates across the entire service, including complex procurement journeys.
- Unlock personalisation for website users via DfE sign-in - saved searches, bookmarks, recommendations and email notifications

This document identifies the conflicts and considerations for this merge.

High-level Technical Architecture

Current Technical Architecture Diagrams




Detailed view of FABS set up
Azure
GHBS Application
- Customer Satisfaction Survey
- Usability Survey
Heroku
Analytics (Google BigQuery)
dfe-analytics-gem
Link to surveys
JSON API
Error monitoring (Rollbar)
Error alerts (Email/Slack)
CMS (Contentful)
Content API
FABS Application (Ruby on Rails)
Background jobs and Cache (Redis)
Contentful web interface
Content editor
End user

Proposed Architecture Diagram for the combined application

`[To be added]`

Domain Name Change

BFYS is at `https://www.get-help-buying-for-schools.service.gov.uk`

FABS is at `https://get-help-buying-for-schools.education.gov.uk`

We want to use the `education.gov.uk` domain for our services. As part of this merge, all the FABS routes will move to the BFYS app but we need to switch the BFYS domain to the one FABS currently uses.

BFYS currently redirects its root and several paths to FABS via `GHBS_HOMEPAGE_URL`:

- Root (`/`) → FABS
- `/dashboard`, `/profile`, `/create_a_specification` → FABS
- `/journeys/*`, `/support-requests/*`, `/feedback/*` → FABS

After the merge, all content will be served from a single application.

Application Performance and Error Monitoring

We use Azure Application Insights and Rollbar (free plan) for BFYS.

We use Rollbar (free plan) for FABS.

We asked to use Sentry in early 2025 but the request has been stuck waiting for approval ever since. We chose to use Rollbar because the free plan is sufficient for our needs and provides us with much better and actionable error monitoring than Azure.

After the merge, we can continue using the current setup for BFYS. It will be simpler since we will only need one Rollbar instance.

Security

As part of the migration, we will also need to move the FABS secrets stored as environment variables on Heroku over to Azure. We already have a similar setup for the BFYS app on Azure, so this should be a simple copying over of the key-value pairs.

---

1. Dependency Version Conflicts

These must be resolved before merging code.

Dependency	BFYS	FABS	Action Required
Rails	~7	~8.1.1	Upgrade BFYS to Rails 8
Ruby	3.3.8	3.4.7	Upgrade BFYS to Ruby 3.4
Redis	~4.8	~5.4	Upgrade BFYS to Redis 5
Sidekiq	~6.4	~8.0.10	Upgrade BFYS to Sidekiq 8

New Gems to Add to BFYS

```
gem "opensearch-ruby"        # Search engine (FABS uses this)
gem "breadcrumbs_on_rails"   # Navigation breadcrumbs
gem "kramdown"               # Markdown parsing
gem "rack-attack", "~> 6.8"  # Rate limiting
```

---

2. Model Conflicts

CONFLICT: `Category`

	BFYS	FABS
File	`app/models/category.rb`	`app/models/category.rb`
Type	`ApplicationRecord` (DB-backed)	`ActiveModel::Model` (Contentful API)
Purpose	Contentful category mirror for journeys	Contentful category for browsing solutions

`Category` has different meanings in the two applications:

- In BFYS, `Category` refers to the business category that frameworks belong to.
- In FABS, `Category` refers to the user-facing category that buying options (solutions) belong to.

These are similar but not exactly the same thing.

When we merge the applications, we'll need to clearly differentiate between the two concepts by using clear names for each.

The naming should be informed via the Information Architecture initiative being carried out.

CONFLICT: `Page`

	BFYS	FABS
File	`app/models/page.rb`	`app/models/page.rb`
Type	`ApplicationRecord` (DB-backed)	`ActiveModel::Model` (Contentful API)
Source	CMS pages stored in DB	Dynamic pages from Contentful

These serve the same purpose and should be combined. It probably makes sense to use the BFYS setup and start storing FABS pages in the DB as well. We have been meaning to cache FABS pages in Redis as a backup against Contentful going down, so this will address that need.

No Conflict (FABS-only models)

These models only exist in FABS and can be moved directly:

- `Solution`
- `Subcategory`
- `Offer`
- `Banner`
- `RelatedContent`
- `ContentfulClient`
- `SearchClient`

Model Concerns

Concern	BFYS	FABS
`has_related_content.rb`	❌	✅
`filter_form.rb`	✅	❌
`has_normalized_email.rb`	✅	❌

`insights_trackable.rb`	✅	❌
`sanitise_meter_numbers.rb`	✅	❌
`task_counters.rb`	✅	❌

No conflicts in concerns.

---

3. Controller Conflicts

CONFLICT: `ApplicationController`

Both have `app/controllers/application_controller.rb`. Will need to merge functionality.

CONFLICT: `PagesController`

	BFYS	FABS
File	`app/controllers/pages_controller.rb`	`app/controllers/pages_controller.rb`
Purpose	Renders DB-backed CMS pages	Renders Contentful pages

No Conflict (FABS-only controllers)

These controllers only exist in FABS:

- `CategoriesController` - might need to be renamed based on what we decide for the models.
- `SolutionsController`
- `OffersController`
- `SearchController`
- `ContentfulWebhooksController`
- `EventsController`
- `Bfys::SolutionsController` (API endpoint)

Controller Concerns

Concern	BFYS	FABS
`breadcrumbs.rb`	❌	✅

`cec_email_template_filters.rb`	✅	❌
`exception_data_prepareable.rb`	✅	❌
`has_date_params.rb`	✅	❌
`has_ticket.rb`	✅	❌
`support_agents.rb`	✅	❌

No conflicts in controller concerns.

---

4. View Conflicts

CONFLICT: `layouts/application.html.erb`

Both have this file. Will need to merge or choose one.

CONFLICT: `pages/show.html.erb`

	BFYS	FABS
File	`app/views/pages/show.html.erb`	`app/views/pages/show.html.erb`

No Conflict: `shared/` **partials**

Both have a `shared/` directory but no file name overlaps.

BFYS shared partials: Only `_start_button.html.erb`

FABS shared partials (can be added directly):

- `_breadcrumbs.html.erb`
- `_content_banner.html.erb`
- `_dfe_base.html.erb`
- `_dfe_footer.html.erb`
- `_dfe_header.html.erb`
- `_dfe_homepage_header.html.erb`

- `_dfe_logo_and_menu.html.erb`
- `_dfe_search_box.html.erb`
- `_dfe_search_button.html.erb`
- `_dfe_search_homepage.html.erb`
- `_dfe_service_navigation.html.erb`
- `_dfe_sidebar.html.erb`
- `_ms_clarity.html.erb`
- `_provider_and_expires.html.erb`
- `_provider_reference.html.erb`
- `_related_content.html.erb`
- `_solution_section.html.erb`
- `_solutions.html.erb`

No Conflict (FABS-only view directories)

- `categories/`
- `solutions/`
- `offers/`
- `search/`

Additional FABS layouts

- `layouts/homepage.html.erb`
- `layouts/all_buying_options.html.erb`
- `layouts/all_offers.html.erb`
- `layouts/pages.html.erb`
- `layouts/search.html.erb`

---

5. Helper Conflicts

CONFLICT: `ApplicationHelper`

Both have `app/helpers/application_helper.rb`. Will need to merge.

No Conflict (FABS-only helpers)

- `ContentfulHelper`
- `GovukDesignSystemHelper`
- `MarkdownHelper`
- `OfferHelper`
- `SolutionHelper`

---

6. Service Conflicts

CONFLICT: `UrlVerifier`

	BFYS	FABS
File	`app/services/url_verifier.rb`	`app/services/url_verifier.rb`

This will not be needed once the applications are combined and can be deleted.

No Conflict (FABS-only services)

- `SolutionIndexer`
- `SolutionSearcher`

---

7. JavaScript Conflicts

CONFLICT: `application.js`

Both have this entry point file.

No Conflict (FABS-only controllers)

These Stimulus controllers only exist in FABS:

- `engagement_tracking_controller.js`
- `external_link_tracking_controller.js`
- `search_toggle_controller.js`

Note: BFYS has `dfe_analytics_engagement_tracking_controller.js` which may overlap with FABS's `engagement_tracking_controller.js`.

---

8. Stylesheet Conflicts

CONFLICT: `application.sass.scss`

Both have this main stylesheet file.

FABS has only the one main stylesheet. BFYS has many additional partials.

---

9. Initializer Conflicts

Initializer	BFYS	FABS	Notes
`assets.rb`	✅	✅	Merge
`content_security_pol`	✅	✅	Merge policies

`icy.rb`			
`dfe_analytics.rb`	✅	✅	Merge config
`filter_parameter_logging.rb`	✅	✅	Merge
`inflections.rb`	✅	✅	Merge
`rollbar.rb`	✅	✅	Same config likely
`sidekiq.rb`	✅	✅	Merge
`contentful_initializer.rb`	❌	✅	Add to BFYS
`rack_attack.rb`	❌	✅	Add to BFYS

---

10. Route Conflicts

FABS Routes (to be added)

```
root "categories#index"

resources :categories, only: %i[show], param: :slug do
  resources :solutions, only: %i[show], param: :slug, path: ""
end

resources :categories, only: %i[index], param: :slug
resources :solutions, only: %i[show index], param: :slug
resources :offers, only: %i[index show], param: :slug

resources :contentful_webhooks, only: %i[create]
post "delete_contentful_entry", to: "contentful_webhooks#destroy"

get "/search", to: "search#index"
post "/events", to: "events#create"

namespace :bfys do
  resources :solutions, only: [:index]
end

get "/:slug", to: "pages#show", as: :page  # Catch-all for dynamic pages

```

BFYS Conflicts

- **Root route**: BFYS redirects to `GHBS_HOMEPAGE_URL`. Change to `categories#index`.
- `/:slug` **catch-all**: BFYS has `get ":slug", to: "pages#show"` at line 646. Both serve dynamic pages - need to merge or prioritize.

---

11. Locale Conflicts

File	BFYS	FABS
`en.yml`	✅	✅

Will need to merge translation keys.

---

12. Error Classes

FABS has `app/errors/contentful_record_not_found_error.rb`. BFYS doesn't have an `errors/` directory - this can be added directly.

Implications for Analytics

We use DfE analytics in both applications. When we merge them, the analytics may also need to be merged. We'll need to **work closely with our performance analyst to plan the best way forward**.

Currently, each application has its own configuration and writes event data to its own tables in Google BigQuery. FABS does not have a database in the application so the configuration is simpler.

BFYS does have a database so there is configuration to exclude sensitive database fields.

We'll need to make sure when we combine the apps, historical analytics data is not lost or corrupted and new data for the combined app makes sense in terms of BI reporting needs.

Implications for Testing

Practical Approach - Use Feature Flags

We want to do the migration gradually in small pieces, testing everything continuously to give us the highest level of confidence in our plan.

The simplest way to achieve this is to use feature flags, which we already use in the BFYS application (via the Flipper gem). We can turn on the GHBS repo routes and pages only in our development and test environments.

This avoids having to do one big merge or massive change in one go, which risks breaking things and introducing unforeseen bugs.

Regression Testing Focus

A key risk during the migration is unintentionally breaking existing, working functionality while making infrastructure or configuration changes that appear unrelated to application behaviour.

To mitigate this, regression testing will be a core part of the migration approach, not a final validation step.

As each change is introduced, whether infrastructure, configuration, routing, environment variables, or deployment pipelines etc we will run regression tests to ensure that existing user journeys, integrations, and critical business flows continue to behave as expected.

This allows us to quickly identify whether a change has introduced a regression and to isolate the cause while the change set is still small and easy to reason about. This in turn reduces risk compared to a big bang approach.

Automated regression tests provide fast, repeatable feedback and give us confidence to move forward incrementally. They also allow us to safely refactor or reconfigure components during the migration, knowing that we have coverage over the most important functionality. Where automation coverage does not yet exist, targeted manual regression testing will be used to reduce risk for critical paths.

By embedding regression testing into each stage of the migration, rather than relying on a single end-to-end test phase, we significantly reduce the risk of late-stage surprises and ensure that the application remains stable and usable throughout the transition to Azure.