InterEd Admin Portal Security Architecture

Role-Based Access Control (RBAC) Implementation

Super Admin Regional Manager Counselor University Relations Finance Team

Complete system Full regional Student-facing Institution Payment and

access access operations management financial access

Y

Permission Hierarchy

| Resource-level Action-level Context-aware Time-based access |

Y Y

(- i" (-

Data Encryption Strategy API Security Measures

At Rest In Transit JWT Tokens Rate Limiting Input Validation Security Headers

Stored data HTTPS/TLS 1.2+ Secure auth 600 req/min All endpoints HSTS, CSP, ete.

—\_ r ‘..

Y Y

Audit Logging Implementation

Authentication fens | DataMoieaons | | Adminacions | | Sytem AccssLogs || Err Tracking

InterEd Admin Portal Integration Architecture

Integration Patterns

= REST API

Primary integration method

+ Webhook Events

Real-time notifications

InterEd Admin Portal

(MERN Stack)

API Gateway / Integration Hub

Security Measures

\* API Keys

For external service auth

« OAuth 2.0

For third-party integration

« Batch Processing

High-volume data exchange

API Gateway

Authentication, Rate Limiting, Request Routing, Transformation

Data Synchronization Layer

= Data Encryption

For all external connections

| HMAC-SHA?256 {| Retry Nv

Synchronization Mechanisms

i ‘

University Portal Intégrations Payment Gateway Integrations

UCAS Integration QUAC Integration Stripe PayPal

UK Applications Canadian Applications Application Fees International Payments

EU Application Direct University Flywire Local Payment

Systems APIs Education Payments Methods

Ne : -, y 4 i \_/

re ~ fc =

Webhook Implementation Third-Party API Integrations

c : , ( : ?

Event T Communication

Real-time Scheduled Event-driven Manual SendGrid, Twilio

Stidantcsaied | applicatior Socket.io Periodic jobs PubSub User-initiated :

Immediate Batch Message On-demand

( — updates processing queue sync Video Conferencing

sty = pices g

L y Calendly, Google Zoom, Teams

ee /

InterEd Admin Portal Performance Optimization Strategy

Performance Targets

+ Page Load: < 2 seconds

\* API Response: < 500ms

InterEd Admin Portal

High-Performance Architecture

>

API Response Time

Monitoring Tools

+ ELK Stack

+ Sentry.io for Error Tracking

Y

P{ Time to imeracve ries Memory Usage ( Network Requ

~ f

Caching Strategy Pagination Implementation

Server-side Caching Client-side Caching Server-side Pagination Client-side Components

| Redis Cache | | React Query Caching | | Limit-Offset Pattern | Infinite Scrolling

Cached API Responses Local Storage Cache Default Page Size: 20 Traditional Pagination

(TTL: 5-15 minutes) (User Preferences, UI State) (Configurable up to 100) (With Page Jumping)

/ M

Y

Y

Data Loading Patterns

Lazy Loading

Prefetching

| Cade Splitting

| Hover Intent Prefetch |

On-demand Loading

(Routes, Components, Data)

Predictive Data Loading

(Based on User Patterns)

Resource Optimization

Frontend Resources

| Bundle Optimization |

Image Optimization

(Compression, Sizing, CDN)

~,

r

Backend Resources

| Memory Management |

Connection Pooling

(Database, External APIs)

InterEd Admin Portal DevOps Pipeline

Continuous Integration / Continuous Deployment Workflow

Che | red | Authillted | ay | Autdifated | monlllbing se

Commit Process Tests Deployment Feedback

Environment Configuration

Development QA/Testing Staging Production

Doacker-based Local Env Isolated Test Environment | Production-like Environment | Highly Available Infra

Continuous Integration Test Data Management UAT Testing | Load Balancing

| Feature Branches | | Automated Test Suite | | Performance Testing | | Auto-scaling |

Deployment Strategy

CI/CD Tools Deployment Techniques Disaster Recovery

GitHub Actions | Blue-Green Deployment

Docker Containerization Canary Releases

| Kubernetes Orchestration Automated Rollback Capability ||

Config Server )

Centralized Config ]

InterEd Admin Portal Microservices Architecture

API Gateway )

Centralized Access Point ]

[ Service Registry

Service Discovery

+ User Authentication

+ JWT Management

\* Role Authorization

» Access Contral

Authentication Student

Service Management

Responsibilities: Responsibilities:

» Student Profiles

» Academic History

= Student Lifecycle

« Status Management

Agent Application

Management Processing

Responsibilities: Responsibilities:

» Agent Profiles

= Performance Tracking

« Commission Mgmt

» Sub-agent Networks

\* Application Workflow

» Status Tracking

+» Offer Management

\* Visa Processing

Document

Management

Responsibilities:

« Document Storage

+ Verification Workflow

+ Version Control

+ Document Processing

f \yf ‘ \yf Vf

Communication Payment Analytics & University Content

Service Processing Reporting Partnership Management

Responsibilities: Responsibilities: Responsibilities: Responsibilities: Responsibilities:

= Email Notifications = Fee Collection + Business Intelligence = Institution Database \* Marketing Resources

« SMS Messaging « Commission Payments \* Dashboard Metrics « Program Catalog \* Email Templates

» Chat System « Refund Processing » Custom Reports « Agreement Tracking « Media Management

« Notification Center + Financial Reporting + Data Visualization + University Relations » Content Distribution

+N \_/ 7 “| a +.

re

Inter-Service Communication Patterns

[ RESTful API Requests

| | Message Queue (RabbitMQ) |

Fault Tolerance Strategies

[ Circuit Breaker |

Retry Mechanism

[ Fallback Se vices |

InterEd Admin Portal Real-time Communication Architecture

Architecture Benefits

+ Low Latency Updates

+ Cross-portal Communication

\* Scalable & Resilient Design

. i- mT cume ni i

Socket.io Server

Real-time Communication Hub

Portal Connections

Technology Stac

= Socket.io (Client/Server)

k

\* Redis for Pub/Sub & Scaling

\* MongoDB Change Streams

\* React for Real-time Ul

Admin Portal

Staff Dashboard

Student Portal

Applicant Interface

Socket.jo Client

Socket.io Client

~

Socketio Implementation

Connection Management Namespaces & Rooms

| Automatic Reconnection | | Portal-specific Namespaces

| Transport Fallback | | User-specific Rooms |

= Connection Authentication L = Application-based Rooms L

v A

~

Notification System Design

Notification Types

Status Updates ]

Task Assignments

Messages & Communications

Delivery Channels

In-app Notifications ]

Email Alerts

SMS/Mobile Push

Agent Portal

Recruitment Partners

Socket.jo Client

fr ~

> | Event-Driven Agshifectu re

Event Producers Event Consumers

Application Services UI Components.

User Action Triggers | Notification Service |

External System Events Logging & Analytics

EY ae U |

a “

Real-time Updates Flow

InterEd Admin Portal Internationalization Architecture

Target Regions Priority Languages

i18n Framework

+ India (Primary) \* English React-Intl / i18next | English (Default) Hindi |

\* Middle East \* Arabic : ;

« East Asia « Chinese Temi Arabic

t a 1

rf a

Language Selector UI Component: | English fect L Sub | au ysl { IZ | Francais

i18n Implementation Strategy

fr ~~ f ss f sy f ™

Translation Management Component Architecture Language Detection Performance Optimization

\* Centralized JSON files \* HOC wrappers » Browser preferences + Lazy loading translations

\* Translation workflow \* Internationalized hooks « User profile settings « Code splitting by language

\* Dynamic content loading + Context providers « |P-based geolocation \* Caching strategies

+ Automatic extraction + Message formatting + Manual selection Ul \* Bundle size optimization

N. ag. /\*. / \* ¥ /

fr Y “y i

Content Management for Multiple Languages Date/Time/Currency Formatting

Content Structure Translation Workflow Date & Time Handling Currency Handling

Key-based Organization Professional Translation [ Intl.DateTimeFormat } [ Intl. NumberFormat }

[ Namespaced Content ] [ Localization Review ] [ Time Zone Management } [ Currency Conversion

7 Variable Interpolation Version Control L Calendar Localizations Regional Price Formatting

F

RTL Support Implementation

CSS & Layout

| | Component Adaptation | | Text Direction Handling | | RTL Testing Framework

InterEd Admin Portal User Journey Maps

Critical Admin Workflows

< Start Point ) C End Point )

Task Step

1

| Decision Point t

System Response | —— Flow Path

1. Application Processing Workflow

Happy Path

Error Path

Wes Decision Path

Review New is Complete? Verify Is Eig?

Application Documents System Response

s No

: ' : V..-...-. '

| Incomplete Application ' i Not Eligible '

| Fone | ce 2 ite a RENO oe | Siento Wee ?

2. Agent Performance Management Workflow

. No

Review Agent Performance Issue?

Dashboard

fe

Schedule Call

Analyze

with Agent

Conversion Rates

Bu

Fe

Improvement Plan? (ene

l A |

te

[create

3. Document Verification Error Recovery Path

InterEd Admin Portal: Comprehensive Architecture

Database Schema

MongoDB Collections

Relationship Modeling

Data Flow =

API Request/Response ee

State Management Flow

Caching Strategy

Redis Server-side Cache

Client-side Optimizations

Performance Optimization

Query Optimization

Resource Management

DevOps & Infrastructure

MongoDB

(~ “) f ="

Frontend Architecture Backend Architecture

r r r — —

Component Architecture State Managemen’ API Gateway Microservices

React Components Redux Store Centralized Access Point Service Boundaries

Modular Design Action Patterns 3 Request Routing Inter-service Communication

" ta

i Fi pg : =

Responsive Design ionalizatio %, Authentication Service Real-time Communication

a

Mobile-First Approach guage Support Security @Access Control Socket.io Implementation

Adaptive Components abiliti Core InterEd JWT Ma jement Event-driven Architecture

Admin Portal ax, - f

nS ‘ [ \ Services / wy

a 4

Deployment Pipeline

Cl/CD Workflow

GitHub Actions

-

Environment Configuration

Dev, QA, Staging, Prod

Configuration Management

Monitoring & Alerting

Performance Tracking

Error Reporting

Deployment Strategy

Blue-Green Deployment

Rollback Capabilities

Cross-Cutting Concerns

a c ‘) » a »

Security Architecture Integration Architecture User Experience Architecture Advanced Capabilities

Role-Based Access Control University Systems User Journey Maps Real-time Communication

Data Encryption Payment Gateways Error Handling Fault Tolerance

API Security Measures Document Verification Internationalization Performance Optimization

Audit Logging Communication Services Accessibility Testing Strategy

he \, Zs \ r = J

InterEd Admin Portal Responsive Design System

c

Breakpoint Strategy

Mobile Tablet Small Tablet Large Desktop Small Desktop Large

I. a S76px - Re sen nos A2Qny - 1199p» 2500

CSS Implementation 2-colum Mobile-First Approach ull layout Component Examples

layout th sideba}

Media Queri Design for mobile first, then progressively Dashboard Card

eda Queries enhance for larger screens

| @media (min-width: 768px) {... } \ “ Desktop Mobile

. Full featured KPls only

Component Adaptation Patterns eh tune oe swultos view

CSS Modules

a mponent Collapse Priority Content Progressiy Table

Full components Primary content Informa Desktop Mobile

Tailwind CSS collapse into displayed first, on-de All columns Cara-based list

. . . pagination key fields only

| Utility-first approach for flexibility compact versions secondary hidden accordi

\ Z MS

a ~ f~

Mobile-Specific Optimizations Layout System Architecture

Touch Targets Gesture Support Grid System

+ Minimum 44x44px hit areas « Swipe navigation 12-column fluid grid with gutters that adjust by breakpoint

+ Adequate spacing between « Pull to refresh

« Interactive elements \* Pinch to zoom ° .

Layout Components Spacing System

Performance Navigation ; ; ;

+ Page Container + Consistent spacing scale

\* Reduced image sizes « Hamburger menu pattern « Section Container « 4px base unit

« Minimal animations \* Bottom navigation bar \* Card Layout \* Responsive margins

q + Network-aware loading \* Floating action buttons © « Split Layout + Auto-adjusting padding

}

Mobile — Tablet — Desktop

InterEd Admin Portal - Data Flow Architecture

Frontend Components

Flow Legend

— User Input

tte State Update

— API Request

=a Realtime

vee Error Flow

( 7) ) )

User Input :

——— Redux Store API Client

State Slices ie Supabase Client

Seen ake Update Actions/Reducers: a aan on eee Auth Client

L | Ow shin | Storage Client j

sant Eo Update State

c “) ae c a ‘ “

Error Handlers Asyne Thunks API Call Realtime]Listeners

Error Boundaries API Requests | Realise Supabase Sfibscriptions

Error Ul] Components Side Effects WebSoi Handlers

h \ #

@

@

Backend (Supabase)

Flow Sequences

©

@

©)

AP! Request Flow

Realtime Flow

Error Flow

Key Processes

Form Submission

1. User input validation

2. Redux thunk dispatch

3. API client request

4, Success/Error handling

5, State update

Realtime Updates

1. Subscribe to changes

2, Receive websocket event

3. Parse payload

4, Update Redux store

5. Ul rerender

Error Handling

InterEd Admin Portal - API Architecture

Client Side (React/Redux)

Supabase Backend

Database

Auth

Storage

©OOC0CO

Realtime

Integration Points

Pid

API Flow Patterns

e

Standard CRUD Pattern

Optimistic Updates Pattern

Realtime Subscription Pattern

// Service layer function

export const getStudents = async (filters = {}) => {

let query = supabase.from(‘students’).select('\*");

// Apply filters

if (filters.status) {

query = query.eq('status’, filters.status);

const { data, error} = await query;

// Redux thunk with optimistic update

export const updateStudent = createAsyncThunk{

‘students/updateStudent’,

async (student, { dispatch, rejectWithValue, getState }) => {

// Save original for rollback if needed

const originalStudent = getState() students

.entities(studentid];

try {

// Optimistically update Ul

dispatch(studentsAdapter.upsertOne(student));

// Setup realtime subscriptions

export const setupRealtimeSubscriptions = (dispatch) => {

// Subscribe to application status changes

const subscription = supabase

from('applications')

on(‘UPDATE’, (payload) => {

// Dispatch to Redux

dispatch(updateApplication(payload.new));

// Trigger notifications if status changed

“a Feron throw ten AR ETronerror message, eror.codey-

return data:

i Actual API call

if (payload.new.status !== payload.old status) {

“dispatchtaddittotificattor Tt

message “Application status undated to $inayload new status)” J

i

const result = await studentsService.updateStudent(student);

ys

Error Handling Strategy

[ Centralized Error Handling

JI

Error Classification & Retry Logic |

Caching Strategy

Query Result & Entity Caching

~,

InterEd Admin Portal - Database Schema Architecture

Database Schema

id UUID (PK) id UUID (PK) id UUID (PK) id UUID (PK)

email TEXT (UNIQUE) first\_name TERT Pe student\_id UUID (FK) university id UUID (Fk)

role\_id INT (FK) last\_name TEXT program \_id UUID(FK) = Le \_| name YEXT

created at TIMESTAMP q email TEXT (UNIQUE) J agent\_id UUID (Fk) level TEXT

updated\_at TIMESTAMP aN. phone TEXT i status TEXT tuition\_fee NUMERIC

Indexes: email el status TEXT : submission\_dat#lMESTAMP status TEXT

ALS: Only self or admin access ‘agent\_id UYID (FR) | | decision date TIMESTAMP created\_at TIMESTAMP

i created \_at TIMESTAMP i created\_at TIMESTAMP. updated\_at TIMESTAMP

F eoles updatedsat — TIMESTAMP updated\_at TIMESTAMP indexes: university id, level, Status J

icuenerr erat gent idy status Indexes: student\_id, program\_id, agent\_id, status /

id INT (PK)

name TEXT (UNIQUE)

created\_at TIMESTAMP

name 7” TEXT

ALS: Read-only for all Zz

id UUID (PK) we id UUID (PK) ee

\*s file-path TEXT

name TEXT ns name TEXT — fb ee 5

7 eas entity\_type TEXT

email TEXT (UNIQUE) ~<| country TEXT aes

See entity\_id UUID

status TEXT statis~ \_ TEXT Car

ae verification\_status TEXT

type TEXT website Sa. eT

: Soe & created\_at TIMESTAMP

parent\_agent\_id UUID (FK) created\_at TIMESTAMP~~.\_\_

Tt esas L.created\_by UUID (FK)

created\_at TIMESTAMP updated\_at TIMESTAMP

indexes, entity type, entity id, verification status |

q Indexes: name, country, status

updated\_at TIMESTAMP

indexes: email, status, parent\_agent\_id

RLS: Agent hierarchy access control Legend

08 User Managenfft Student/Agent

Performance Optimization: indexes on foreign keys, frequently filtered fields (status), and unique fields (email) |B Applications Universities/Pro. he

‘Za ™~ f~ ~

Row Level Security Policies Data Validation & Migration Strategy

+ Email format validation \* Version-controlled migrations

+ Status field enums + Up/down migration seripts

+ Required field constraints + Sequential numbering

+ Foreign key constraints + Transaction-based changes

\* Numeric range validation \* Seed data management

+ Timestamp validations + CI/CD pipeline integration

InterEd Admin Portal - Deployment Architecture

CI/CD Pipeline

Deployment Environments--

\*

Development Environment

Staging Environment

Production Environment

Backend

Supabase Dev

Frontend

Vercel Preview

Frontend

Vercel Staging

Backend

Supabase Staging

Frontend

Vercel Production

Backend

Supabase Production

os

Configuration

Development Environment Variables

Feature Flags for Development

Configuration

Staging Environment Variables

Praduction-like Settings

Configuration

Production Environment Variables

Optimized Performance Settings

‘Automatic deployment for feature branches

Automated deployment from main branch

Manual promotion fram staging

Developer-specific preview environments User acceptance testing environment Production database with backup strategy

. £

Monitoring & Logging Rollback & Recovery Strategy

r a ‘) ; F i "

Application Monitoring Logging System Version Control Rollback Database Rollback

\* Vercel Analytics \* Structured Logging + Git revert and rollback procedures. + Supabase point-in-time recovery

+ Performance Monitoring + Log Aggregation + Previous release tagging + Migration down scripts

\_\_\_+ Real User Monitoring (RUM) + Log Retention Policies J + Hotfix branch strategy (\_\_+ Database snapshot strategy

r i, ;

Error Tracking Alerting System Deployment Rollback Disaster Recovery

\* Frontend Error Capture + Threshold-based Alerts \* Vercel instant rollback + Full system backup strategy’

+ Supabase Error Lags + Notification Channels (Email, Slack) + Previous deployment activation + Recovery time objectives (RTO)

\* Error Notifications + Incident Response Procedures + Environment-specific rollback | | «Incident response playbooks

r, \.

InterEd Admin Portal - Feature Module Architecture

Core Infrastructure

se

Feature Modules

Student Management Module

-

ten Ped

StudentFormComponents

\* studentThunks

« studentActions -

StudentListPage Feature State ApplicationListPage Feature State

A

+ studentsSlice + applicationsSlice !

StudentDetailPage Seiideaitalactoe ApplicationDetailPage + applicationSelectors

St iinininnecan

i

\* applicationThunks a

\* applicationActions ri

é

ta

studentService

applicationService

mi

AgentListPage

T

Feature State ri

1

T

1

‘

UniversityListPage

Feature State

a

AgentDetailPage

\* agentsSlice ’

ui

T

v

|

+ universitiesSlice

UniversityDetailPage

,

\*agentSelectors | = programsslice

-

\* agentThunks ~ + universitySelectors

AgentFormComponents a agentactions ProgramManagement » university hunks

agentService universityService

Depends on: Auth, Documents

Required by: Applications

Feature Flag System

Feature Flag Configuration

+ Environment-based flags

+ User-role based flags

+ A/G testing capability

Feature Implementation

\* Conditional rendering

+ Route-based feature gates

+ AP! feature flags

\* Feature toggle hooks

Ready for 48-hour hackathon:

Simple enviranment-based JSON config

Applications module has the most dependencies

No[eqgeHa’ activation controlled via feature flags

Core Infrastructure

---- Module Dependel\_tsintegration Points

Student Features

Agent Features

Application Features

University Features

Feature

Flag System

InterEd Admin Portal - Error Handling Architecture

~ if" = f"

Error Boundary Strategy Error Classification API Error & Retry Handling

‘ f “) c

App-Level Error Boundary Frontend Rendering Errors Error Response Handling

« Last line of defense \* Component lifecycle errors \* Centralized error handling middleware

+ Prevents white screen of death J \* Prop tyne mismatches + Error normalization across APIs

Route-Level Error Boundaries API Request Errors .

Retry Strategies

+ Isolate errors to specific routes \* Network failures (4xx, 5xx) 1. Error ofcurs ;

a . er woeeebhot-“B=4 = \* Exponential backoff for network errors

\\_+ Allow navigation to working routes + Authentication errors eee :

ae --" \* Maximum retry attempts configurable

}) + Validation errors were? No retry for 4 (client

‘ 4 « NO re Tr 4xx (client) errors

Feature-Level Error Boundaries P ‘ =

+ Automatic retry for idempotent requests

x

L

« Isolate errors to specific features Data Handling Errors ;

-

‘

\\_\_+ Preserve functionality of other features \_\_J . ee 1 2. Retry mechanism

; = + Data transformation errors Circuit Breaker Patter

' .

For critical components:,DataTable, Forms, Charts + Sldie managenmentesceptions 2 A . Ma

pe 1D f Prevent cascading failures-\*

Enables fine-graingd fallback strategies Severity: Critical, High, Medium, Low \_4 + Automatic service degradation

1 al

fal component tails F, “a

- aa

' a £ 3, Log ergy” 1 £

Fallback UI Strategy Error Logging & Reporting ,”” User Notification Patterns

é

. ee

‘ ¢

r ae “ r

. es se: .

Generic Error States Frontend Error Logging Toast Notifications

\*

\* App-level crash screen \* Global error event listener” « Transient error messages

+ Route-level error page + React error boundary reporting + Auto-dismissing for non-critical errors

\* Feature module error state + Console errors capture | \_ 4 ‘

+ Error with support contact information-©9 COMPONENt erfor iB = 5 P

ae J a a Inline Error Messaging

=. SSSSosceeey 4 -- P= BackentrError in Sh Ce a WO cas

teggin 9 >'Form field validation errors

Component-Specific Fallbacks 6, Report resolution "= ~==

\* Supabase error lagging w= 42) sComponent-specific errog states

7

+ Empty state data table \* Database query errors = \* § User action

4

\* Form validation errors wi,

porary f } Modal Errar-Dialogs

+ Chart/visualization fallbacks ;

: . Error Reporting System = =

+ Skeleton loaders during retries + Critical errors requiring acknowledgment

+ Partial data display with error indicators \* Structured error format + Errors with action options (retry, cancel)

\* Error aggregation by type/component

+ Error frequency tracking [ Persistent banners for system-wide issues

\ J

y, \. A \.

Error Handling Implementation Priority for 48-Hour Hackathon: Critical Path First

InterEd Admin Portal - Testing Architecture

Test Types & Scope

Unit Tests Integration Tests End-to-End Tests |

\* Individual components, functions, hooks + Feature module integration \* Critical user flows

+ Redux slices, reducers, selectors Jest + Form submissions with validation Jest + Authentication flows Cypress

+ Utility functions React Testing Library + AP| service integratideact Testing Library + CRUD operational kepebbtiemstance

» Service modules 75-85% Coverage Goal + Redux store with Br6Ptineoterage Goal + Multi-stef:qiioa=paths only (10-15 tests)

H H \ H

T T T

Implementdinit tects first : : -

' i '

t i i

2 \*, ‘i 4 + i i ‘5

< - \* . Mock ent

Test File Organization Mock Strategies | “Orn” Test Data Management

!

ab L 4

= > 1 ( :

Co-location Strategy Component Mocking Test: Fixtures

T

« Unit tests co-located with source files \* Mock child components when testing parents + JSON fixtures for entity data (students, agents)

te at Pare i ” Paes: Cree \* ce a! +

» Componentjsx and Componenttest,jsx + Use Jest's manual mocks for complex components \* Mock API responses int cqneistent (Oat tom h elpers

+ Makes tests discoverable and maintainable + Focus on component interface, not implementation + Shared across test suites for consistency J

= d T

aya i 1

Tests Directory Structure Service Mocking ONE Factory Functions

= L

| 1

+ /src/components/ComponentName/ComponentName.test jsx \* MSW (Mock Service Worker) for AP] mocking |" ~} =. Functions to generate fest data dynamically

+ /src/utils/utilName.testjs \* Mock Supabase client in tests + Customizable entity fattories (createStudent)

+ /eypress/integration/features/featureName.specjs. + Response fixtures for different scenarios + Reduces test data maintenance

« Test success, error, and loading states: + Ensures valid entity relationships

a, M J a” J

Consistent naming: Component.test,jsx pee

7 Integration tests: feature-name.specjsx ji ( ger 1 i

PasiWwe COVER \*

= Tésting Utilities & Helpers

\_ a? i

Custom Render

+ Extends React Testing Library's render

Rid A = oa

- - Testing Hooks

i + Custom renderHook with providers

48-Hour Hackathon Testing Focus =e =

= Fest Redinchooks-wittrstore comext

Focus on unit tests for critical components and utility functions f q

on . ms i ; User Interaction Helpers

Prioritize tests for core business logic and data transformation J

+ Form filling helpers (fillForm, submitForm)

| —rtabteinteractior-hetpers-tsorttabte-fittertabtey ——"\_\_ ]