Testing

.UnitTests

Test that individual pieces work in isolation for given I/O

Mock any dependencies

.InternalIntegrationTests

Test that libraries interact in expected ways

.ExternalIntegrationTests

Test that app interacts with external systems as expected

FUTURE

Back-End

.Persistence

Saving and loading .Core objects

- Uses Entity Framework Core

- .Web.Startup configures SQL
Server conn

.Core

Business logic and models

- Classes: Task,

.Workflows

Custom Elsa classes for:

Activities

FTP

Files (Copy & Move)

PowerShell

SQL

Workflows (Code-Defined)

.Reporting (Future)

Custom code for logging,
notification via email/chat/etc.

Custom Activity-supporting Libraries

.FTP

Provides IFTPService

Wraps WinSCP API -
Must be installed on
host

.PowerShell

Permissions determined
by user running app -
need to figure out pass-
through or impersonate

.Web

All startup, config,
DI happens here

API

Pages

Front-End

Elsa Dashboard

Visual workflow build & management

wkbenslinux.rbc.org:14000

MUST figure out auth
to run these separately

Via link from Pages
in Single Task View

User

Front End
Mockup

Blazor Pages

Options & Downsides

Blazor Server

Active connection, user must
reload if connection is lost unless
handling deliberately included

Blazor WASM

Bleeding edge, less 3rd party
library support

.NET Core runtime for browser
download on first load

Person

- Name
- Phone Number
- Email Address

- Purchase Parking Pass

Address

- Street
- City
- State
- Postal Code
- Country

- Validate
- Output As Label

0..1 — lives at — 1

Student

- Student Number
- Average Mark

- Is Eligible To Enroll
- Get Seminars Taken

Professor

- Salary