# Initial priority

* Get tests in place so that you’re free to refactor but as long as the tests pass. This will include CICD work to ensure tests are run in the background to keep info current.

# General

* Jira or something to manage tickets? Probably won’t need it, but would be good to get used to linking commits to tickets
* take “count” of stats and work to reduce these. Eg, lines of code – refactor to reduce redundancy, track the results
* books like code construction, apply
* when a version is “perfect”, use as foundation for variants (eg blazor-styled webcomponents, vuejs, nodejs, etc, rust, f#)
* Setup proper CICD (tests running on local gitlab instance, pushes to github, etc)
* Git techniques (compare sourcetree to fork, etc)
* Diff tool (meld?)
* Buy USB scanner gun
* Azure Data Studio
* Postman
* VSCode
* Docker/Kubernetes
* The branding idea – could make a futuristic game-based one? Eg Samus etc.
* Create a public demo website that’s always live.

# VS Code, Top Level

* Advanced powershell – gui for running actions?
* Github workflows = Run production pipeline to release to production test environment?
* Have at least 2 sites, A-B. When deploying to a site, have a site-is-down message
* Azure – turn off during certain times of day
* Reorg solution so that all Req files, pdfs, etc is under a specific folder (eg there’s a qrcodesamples folder and a reqs folder
* Consider: renaming solution to something generic and WoodseatsScouts as a branded implementation
* Sort out Utilities folder
* Powershell via C# - this was problematic, eg reporting PS errors
* Deployments via scripts – updating the connection string was annoying. Not needed unless there was a migration needed, which for a test site wasn’t the case.
* Instead of deploying acceptance test data (eg Glaucous Jet), deploy actual users in addition to AI users (get a service to generate several dozen AI users in the same theme)
* Improve the TodoReport and incorporate it into the dev workflow
* Rider shortcuts, commands
* GitHub wiki
* There’s a “global.json” file in the root, what is this?
* Utilise azure keyvault?

# API / Rider

* Trivial bug: one of the toasts still show ~“The code ‘code’ does not exist.” Should be like the other messages
* Important: get rid of redundant id field in Coins (BaseValueId)
* Learn latest debugging techniques
* Is there a need for an Acceptance Test solution? Specflow? (ie not the playwright stuff)
* Get it running on linux just as a test?
* Latest equivalent of ghostdoc?
* Split up the members table? Others?
* Move interfaces like IAppDbContext to Abstractions
* Performance tests
* Auto build incrementing
* Put ef reports code into t-sql? – Use sql book against project for ideas
* Create helper views?
* Automapper with a gateway shim?
* Get warnings, issues down to zero
* Break up view folder into area – speaking of that, establish proper “Domain” areas. Is this a good case for redoing DDD and UML?
* UserData / MemberImages – move out into a .net core self-hosted “CDN”
* Get serilog logging in the app
* Study how to simply tests, eg Mock, patterns, xunit
* Sharing code standards properly (rider vs stylecop vs etc)
* Is it worth creating resx files for strings?
* Capture usage stats (eg audit, most used page, etc)
* Async everything (required: debugging techniques)
* Packages to study
  + .net 8 web api in general, c# 12
  + .net dependency injection
  + EF Core
  + Microsoft.Extensions.TimeProvider.Testing
  + Microsoft.Powershell.SDK
  + Xunit
  + Moq
  + Shouldly

# React / Webstorm

* Use configurations for different things, eg, in “debug” mode, disable auto-page-refreshing.
* Don’t call window.location.reload, reload data the react way (prevents page flashing)
* Packages
  + React jest
  + Axios
  + Bootstrap, reactstrap
  + Qr-reader, countdown, webcam
  + Toastify
  + Web-vitals?
  + React-scripts
* Nextjs?
* Typescript etc
* “Dev mode” setting, eg helpers like the modal to insert coins
* Reorg solution
* Webstorm shortcuts (eg clean up)
* Local storage
* Console logging (Unicode stuff for example, find package)
* Use emberjs
* Css, Sass!
* Redux
* Audio in browser
* Coin animation? Like an initial flip with a sparkle?
* Spinner is circular, doesn’t fit with theme, if rectangular, make rounded corners
* Rename homepage to coins, default route

# Playwright / Aqua

* Autoscreenshotting / “Actions” pattern (where the method call will output the image and stamp the name on the image), comparisons by branch, generate html page of images (left right arrows to flip between output)
* Reorg
* Find solution to easily stop and start (milestones? Dbsnapshots)
* Speed up tests as fast as possible
* Test across several device emulators
* Proper agnostic acceptance criteria in gherkin – so other variants can reuse the exact same AC to run tests against
* Get playwright to size browsers so they don’t overlap?
* Flaky test – Step 0029 Time’s up
* Autoincremented tests – nice for ordering,but can’t jump to them