(priority, level of effort)

1. Settings Refactoring – see what we can do to clean up implementation, specifically regarding state management
2. Add Settings
   1. (medium, low) Request Timeout
3. Behavior
   1. (high, medium) Async Requests – allow interaction with GUI while request is underway. Allow cancelling request. Submitting a new request implicitly cancels previous request.
   2. (low, low) Prompt to save when trying to close and current request has been modified
   3. (low, low) Swallow exceptions when fail to save settings on window closing (e.g. user local settings have been deleted)
   4. (low, ?) Edit settings validation errors – see if we can use the same kind of error dialog box for custom validation errors that is built into the Settings for data type validation errors. see <http://stackoverflow.com/a/8653764/236255> to use reflection. Note also the behavior differences between OK and Cancel (former focuses back on value in prop gird without changing it, the latter undoes the value just entered).
   5. (low, medium) Implement greater under/redo on request text boxes
   6. (low, high) Implemented auto complete in the request headers text box (both for keys and values)
   7. (medium, medium) Allow adding files to request
   8. (medium, medium) Support file:// protocol (RestSharp doesn’t seem to)
   9. (high, medium) Ensure settings are persisted between versions of app (see <http://stackoverflow.com/questions/621265/can-i-control-the-location-of-net-user-settings-to-avoid-losing-settings-on-app/1928041#1928041>)
   10. (medium, low) Short-cut keys for menu items (e.g. Save)
4. GUI
   1. (low, low) Add splitter between request Url/Buttons panel and body
   2. (low, ?) Remove unused items from text box context menus
   3. (medium, low) Add “?” link next to “Error” response status results which pops up to show error message.
5. Content Output
   1. Rendered
      1. (medium, low) Render media content types, specifically images
      2. (low, low) Resolve relative URLs
      3. (low, low) Open links in the default browser like with Raw and Pretty content
6. Infrastructure
   1. (medium, low) Implement logging – Log4net with simple file logger should be good, will help users to report bugs
   2. Create build script
7. Project Management
   1. Rename
   2. Start google code project
      1. Front page feature overview with screen shot
   3. Import repository
      1. Probably stick with subversion, but maybe mercurial
      2. May or may not care about history
   4. Create user guide
      1. Take snap shots, label GUI elements with numbers, describe
   5. Start free open source Teamcity continuous integration
   6. Register with ohloh (open source project index)