1.1 ./csharp/Platform.Bot/Program.cs

```
using csharp;
using Interfaces;
using Octokit;
using Platform.Exceptions;
using Platform.IO;
using Storage.Local;
using Storage.Remote.GitHub;
using System;
using System.Collections.Generic;

namespace Platform.Bot
{
    /// <summary>
    /// <para>
    /// Represents the program.
    /// </para>
    /// <para></para>
    /// </summary>
    internal class Program
    {
        /// <summary>
        /// <para>
        /// Main the args.
        /// </para>
        /// <para></para>
        /// </summary>
        /// <param name="args">
        /// <para>The args.</para>
        /// <para></para>
        /// </param>
        private static void Main(string[] args)
        {
            using var cancellation = new ConsoleCancellation();
            var argumentIndex = 0;
            var username = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "Username", args);
            var token = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "Token", args);
            var appName = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "App Name", args);
            var databaseFileName = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "Database
            ↪  file name", args);
            var fileSetName = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "File set name
            ↪  ", args);
            var dbContext = new FileStorage(databaseFileName);
            Console.WriteLine($"Bot has been started. {Environment.NewLine}Press CTRL+C to
            ↪  close");
            try
            {
                var api = new GitHubStorage(username, token, appName);
                new ProgrammerRole(new List<ITrigger<Issue>> { new HelloWorldTrigger(api,
                ↪  dbContext, fileSetName), new OrganizationLastMonthActivityTrigger(api) },
                ↪  api).Start(cancellation.Token);
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.ToStringWithAllInnerExceptions());
            }
        }
    }
}
```

1.2 ./csharp/Platform.Bot/ProgrammerRole.cs

```
using Interfaces;
using Octokit;
using Storage.Remote.GitHub;
using System;
using System.Collections.Generic;
using System.Threading;

namespace Platform.Bot
{
    /// <summary>
    /// <para>
    /// Represents the programmer role.
    /// </para>
    /// <para></para>
    /// </summary>
    public class ProgrammerRole
    {
```

```
        /// <summary>
        /// <para>
        /// The git hub api.
        /// </para>
        /// <para></para>
        /// </summary>
        public readonly GitHubStorage GitHubAPI;

        /// <summary>
        /// <para>
        /// The minimum interaction interval.
        /// </para>
        /// <para></para>
        /// </summary>
        public readonly TimeSpan MinimumInteractionInterval;

        /// <summary>
        /// <para>
        /// The triggers.
        /// </para>
        /// <para></para>
        /// </summary>
        public readonly List<ITrigger<Issue>> Triggers;

        /// <summary>
        /// <para>
        /// Initializes a new <see cref="ProgrammerRole"/> instance.
        /// </para>
        /// <para></para>
        /// </summary>
        /// <param name="triggers">
        /// <para>A triggers.</para>
        /// <para></para>
        /// </param>
        /// <param name="gitHubAPI">
        /// <para>A git hub api.</para>
        /// <para></para>
        /// </param>
        public ProgrammerRole(List<ITrigger<Issue>> triggers, GitHubStorage gitHubAPI)
        {
            GitHubAPI = gitHubAPI;
            Triggers = triggers;
            MinimumInteractionInterval = gitHubAPI.MinimumInteractionInterval;
        }

        /// <summary>
        /// <para>
        /// Processes the issues using the specified token.
        /// </para>
        /// <para></para>
        /// </summary>
        /// <param name="token">
        /// <para>The token.</para>
        /// <para></para>
        /// </param>
        private void ProcessIssues(CancellationToken token)
        {
            while (!token.IsCancellationRequested)
            {
                foreach (var trigger in Triggers)
                {
                    foreach (var issue in GitHubAPI.GetIssues())
                    {
                        if (trigger.Condition(issue))
                        {
                            trigger.Action(issue);
                        }
                    }
                }
                Thread.Sleep(MinimumInteractionInterval);
            }
        }

        /// <summary>
        /// <para>
        /// Starts the cancellation token.
        /// </para>
        /// <para></para>
        /// </summary>
```

```
97          /// <param name="cancellationToken">
98          /// <para>The cancellation token.</para>
99          /// <para></para>
100         /// </param>
101         public void Start(CancellationToken cancellationToken) =>
            ↪ ProcessIssues(cancellationToken);
102     }
103 }
```

1.3 ./csharp/Platform.Bot/Triggers/HelloWorldTrigger.cs

```
1   using Interfaces;
2   using Octokit;
3   using Storage.Local;
4   using Storage.Remote.GitHub;
5
6   namespace csharp
7   {
8       /// <summary>
9       /// <para>
10      /// Represents the hello world trigger.
11      /// </para>
12      /// <para></para>
13      /// </summary>
14      /// <seealso cref="ITrigger{Issue}"/>
15      internal class HelloWorldTrigger : ITrigger<Issue>
16      {
17          /// <summary>
18          /// <para>
19          /// The git hub api.
20          /// </para>
21          /// <para></para>
22          /// </summary>
23          private readonly GitHubStorage gitHubAPI;
24
25          /// <summary>
26          /// <para>
27          /// The file storage.
28          /// </para>
29          /// <para></para>
30          /// </summary>
31          private readonly FileStorage fileStorage;
32
33          /// <summary>
34          /// <para>
35          /// The file set name.
36          /// </para>
37          /// <para></para>
38          /// </summary>
39          private readonly string fileSetName;
40
41          /// <summary>
42          /// <para>
43          /// Initializes a new <see cref="HelloWorldTrigger"/> instance.
44          /// </para>
45          /// <para></para>
46          /// </summary>
47          /// <param name="gitHubAPI">
48          /// <para>A git hub api.</para>
49          /// <para></para>
50          /// </param>
51          /// <param name="fileStorage">
52          /// <para>A file storage.</para>
53          /// <para></para>
54          /// </param>
55          /// <param name="fileSetName">
56          /// <para>A file set name.</para>
57          /// <para></para>
58          /// </param>
59          public HelloWorldTrigger(GitHubStorage gitHubAPI, FileStorage fileStorage, string
            ↪ fileSetName)
60          {
61              this.gitHubAPI = gitHubAPI;
62              this.fileStorage = fileStorage;
63              this.fileSetName = fileSetName;
64          }
65
66          /// <summary>
67          /// <para>
68          /// Actions the obj.
```

```
        /// </para>
        /// <para></para>
        /// </summary>
        /// <param name="obj">
        /// <para>The obj.</para>
        /// <para></para>
        /// </param>
        public void Action(Issue obj)
        {
            foreach (var file in fileStorage.GetFilesFromSet(fileSetName))
            {
                gitHubAPI.CreateOrUpdateFile(obj.Repository.Name, obj.Repository.DefaultBranch,
                ↪  file);
            }
            gitHubAPI.CloseIssue(obj);
        }

        /// <summary>
        /// <para>
        /// Determines whether this instance condition.
        /// </para>
        /// <para></para>
        /// </summary>
        /// <param name="obj">
        /// <para>The obj.</para>
        /// <para></para>
        /// </param>
        /// <returns>
        /// <para>The bool</para>
        /// <para></para>
        /// </returns>
        public bool Condition(Issue obj) => obj.Title.ToLower() == "hello world";
    }
}
```

1.4 ./csharp/Platform.Bot/Triggers/OrganizationLastMonthActivityTrigger.cs

```
using Interfaces;
using Octokit;
using Platform.Communication.Protocol.Lino;
using Storage.Remote.GitHub;
using System;
using System.Collections.Generic;
using System.Linq;

namespace Platform.Bot
{
    /// <summary>
    /// <para>
    /// Represents the organization last month activity trigger.
    /// </para>
    /// <para></para>
    /// </summary>
    /// <seealso cref="ITrigger{Issue}"/>
    internal class OrganizationLastMonthActivityTrigger : ITrigger<Issue>
    {
        /// <summary>
        /// <para>
        /// The storage.
        /// </para>
        /// <para></para>
        /// </summary>
        private readonly GitHubStorage Storage;

        /// <summary>
        /// <para>
        /// The parser.
        /// </para>
        /// <para></para>
        /// </summary>
        private readonly Parser Parser = new();

        /// <summary>
        /// <para>
        /// Initializes a new <see cref="OrganizationLastMonthActivityTrigger"/> instance.
        /// </para>
        /// <para></para>
        /// </summary>
        /// <param name="storage">
        /// <para>A storage.</para>
```

```
        /// <para></para>
        /// </param>
        public OrganizationLastMonthActivityTrigger(GitHubStorage storage) => Storage = storage;

        /// <summary>
        /// <para>
        /// Determines whether this instance condition.
        /// </para>
        /// <para></para>
        /// </summary>
        /// <param name="issue">
        /// <para>The issue.</para>
        /// <para></para>
        /// </param>
        /// <returns>
        /// <para>The bool</para>
        /// <para></para>
        /// </returns>
        public bool Condition(Issue issue) => issue.Title.ToLower() == "organization last month
        ↪  activity";

        /// <summary>
        /// <para>
        /// Actions the issue.
        /// </para>
        /// <para></para>
        /// </summary>
        /// <param name="issue">
        /// <para>The issue.</para>
        /// <para></para>
        /// </param>
        public void Action(Issue issue)
        {
            var issueService = Storage.Client.Issue;
            var owner = issue.Repository.Owner.Login;
            var activeUsersString = string.Join("\n",
            ↪  GetActiveUsers(GetIgnoredRepositories(Parser.Parse(issue.Body)), owner));
            issueService.Comment.Create(owner, issue.Repository.Name, issue.Number,
            ↪  activeUsersString);
            Storage.CloseIssue(issue);
        }

        /// <summary>
        /// <para>
        /// Gets the ignored repositories using the specified links.
        /// </para>
        /// <para></para>
        /// </summary>
        /// <param name="links">
        /// <para>The links.</para>
        /// <para></para>
        /// </param>
        /// <returns>
        /// <para>The ignored repos.</para>
        /// <para></para>
        /// </returns>
        public HashSet<string> GetIgnoredRepositories(IList<Link> links)
        {
            HashSet<string> ignoredRepos = new() { };
            foreach (var link in links)
            {
                var values = link.Values;
                if (values != null && values.Count == 3 && string.Equals(values.First().Id,
                ↪  "ignore", StringComparison.OrdinalIgnoreCase) &&
                ↪  string.Equals(values.Last().Id.Trim('.'), "repository",
                ↪  StringComparison.OrdinalIgnoreCase))
                {
                    ignoredRepos.Add(values[1].Id);
                }
            }
            return ignoredRepos;
        }

        /// <summary>
        /// <para>
        /// Gets the active users using the specified ignored repositories.
        /// </para>
        /// <para></para>
```

```
116             /// </summary>
117             /// <param name="ignoredRepositories">
118             /// <para>The ignored repositories.</para>
119             /// <para></para>
120             /// </param>
121             /// <param name="owner">
122             /// <para>The owner.</para>
123             /// <para></para>
124             /// </param>
125             /// <returns>
126             /// <para>The active users.</para>
127             /// <para></para>
128             /// </returns>
129             public HashSet<string> GetActiveUsers(HashSet<string> ignoredRepositories, string owner)
130             {
131                 HashSet<string> activeUsers = new();
132                 foreach (var repository in Storage.Client.Repository.GetAllForOrg(owner).Result)
133                 {
134                     if (ignoredRepositories.Contains(repository.Name))
135                     {
136                         continue;
137                     }
138                     foreach (var commit in Storage.GetCommits(repository.Owner.Login,
                         ↪ repository.Name))
139                     {
140                         activeUsers.Add(commit.Author.Login);
141                     }
142                     foreach (var pullRequest in Storage.GetPullRequests(repository.Owner.Login,
                         ↪ repository.Name))
143                     {
144                         foreach (var reviewer in pullRequest.RequestedReviewers)
145                         {
146                             activeUsers.Add(reviewer.Login);
147                         }
148                     }
149                     foreach (var createdIssue in Storage.GetIssues(repository.Owner.Login,
                         ↪ repository.Name))
150                     {
151                         activeUsers.Add(createdIssue.User.Login);
152                     }
153                 }
154                 return activeUsers;
155             }
156         }
157     }
```

Index