TEAM CHEAPCHAP SOLUTION FOR CODECHEF FLASHFLEX HACKATHON

The solution is towards a browser with adobe support.

Two ways of achieving the above: -

1)To hold to present day browser's version status by not updating the current versions of browsers. Since the problem of mainstream browsers removing the access is done by next introduced version update.

2)To build web browser of today's flash support kind any time after 2020. Mozilla Firefox can be intended to do since it is open source.

Scope of seemed arising problems:

1)What if this is not possible if manufacturer changes terms.

2)What if we lose current version package

3)What if todays version will not be available tomorrow, incase lost.

SOLUTIONS as numbered: -

1)Mozilla Firefox is opensource and we can be authority to own it as our product with future open source leveraged adoptions

2)Even if all is lost, we can easily build from stack and then on customize it for flash enablity.

3)Though we would try to store a copy of them. Though lost in case. The current commit would be still available since its opensource.

Since in the web briefing, it was mentioned in reply to an answer that a subsidiary plugin to support flash would be accepted. This means that procedure follow of end user is at customized our control to this extent. So the ask of using our browser can be done. A directive to adopt firefox browser can be asked for it is quite a minimalist change in client behaviour.

For Example Internet explorer has grown over just in little time from internet explorer to edge to edge chromium recently. This would be an always problem with different authoritative tool usage in our workflow. As a tech intellect provider empowering them by directing them towards an open source leveraged approach they might not be aware off would in actual be a right give to.

With other authoritative tool at place, for all future versions compatable is a false promise we would be delivering, since they might change terms as convience any time. Our now problem to hack is the best example. It is that this dependency of "their authority" policy to remove support for flash is why this discripency. Otherwise in actual works for us so fine.

But with open source adoption, we can actually deliver their promise of future versions compatability too, since we are the authority and can leverage any future changes too.

If the world of internet changes so much in future we can still make it happen as we take that time version and add flash support the subsidary code to be tweaked is also shown ind demo video.

This is actually a right opportunity strategy to adopt for us to keep up with the promise more diligently, which unlike we couldn't fulfilll last time for everlasting compatabile product and regain their trust even subtly.

Something which achieves our goals is the right stack to adopt and can never be obsolete. This is why industry still uses windows xp and before versions at many places, even though a more efficient current versions are available for different tasks. Since as simple it works for them. Similarly flashflex which sufficiently works so fine for us, can be stick to adopt.

"**All these are itterative steps preparing for even more fullfleged, foolproof, no stone unturned protected solution making at each steps. <u>As in actual the first step itself sufficiently solves the problem</u>.**"

Steps to build firefox from its source code:-

Step 1: Clone Mozilla Central

Mozilla Central is the name of the mercurial repository that contains the source code of Firefox. Start by installing mercurial and cloning "MC":

```
hg clone https://hg.mozilla.org/mozilla-central/
```

`hg clone {repo_url}` is the first and most basic mercurial command, but check out my Mercurial Productivity Tips post to learn more `hg` commands! "MC" will be installed in a `mozilla-central` directory.

Step 2: Install Dependencies

From C++ to Rust, Firefox has a fair bit of requirements you'll need to install. From within the `mozilla-central` directory, run the following:

```
./mach bootstrap
```

The `bootstrap` command will install dependencies as well as configure mercurial extensions as required. Congratulations -- you're now ready to build!

Step 3: Create a mozconfig

Create a `mozconfig` file to use artifact builds, which will save loads of time during the build process by downloading pre-built binaries for Firefox's internals.

```
# Automatically download and use compiled C++ components:
ac_add_options --enable-artifact-builds

# Write build artifacts to:
mk_add_options MOZ_OBJDIR=./objdir-frontend
```

Place the code above in your `mozconfig` file and you're builds will be super fast!

Step 4: Build!

Once you have the code and the dependencies, it's time to build the amazing Firefox! You can build Firefox with the following command:

```
./mach build
```

Step 5: Run Firefox

Once you've built the amazing Firefox, you can run Firefox with the following `mach` command:

```
./mach run --jsdebugger
```

Congratulations! You've taken the Firefox source code and turned it into an application that you can run! The `--jsdebugger` option opens the "browser toolbox" which allows you to debug the Firefox you've just built.

The demo video available in the repository proves the following:

1)A current version would be available

2)The way to enable flash enablity in the browser. Where the files of flash disablity to be turned on and in future what scripts can be used to turn it on.

3)Building of firefox from stack.