CATEGORIES OF SURVEY RESULT SUGGESTIONS

INTRODUCTION

Recently, a survey was sent around to Kivy users.

During the Kivy Next live presentations, Mirko (@m1sl6) and I (@julian-o) walked through many of the survey responses.

This is the next step: Looking at the survey suggestions and categorizing them into themes.

Sources of error

Some potential sources of error:

- Surveys are going to collect the opinions of the people who care enough to be monitoring the channels and spend some time on giving their opinion. It isn't going to catch people who tried Kivy and did not stick with it.
- The space given tended to capture quick responses, not detailed explanations, or actionable suggestions.
- I have almost certainly misunderstood the nature of some of the suggestions.
- I copy-edited many answers; there is a chance I unintentionally distorted their meaning.
- Sometimes, a suggestion belonged in multiple categories. I chose the best fit I could.

This should be parsed for its qualitative themes, rather than precise number of respondents,

COMMON ANSWER THEMES

The following sections show the survey suggestions (in green) categorized by themes.

Kivy Development

Kivy Team and Community

- More developers
- Hire more developers using opencollective funds
- onboarding new blood
- increase "bus number" for business continuity
- Kivy community consists of solo/beginner developers. Need enterprise apps users.
- maintain and release man power
- No jobs in Kivy and no developers knowing Kivy - vicious cycle.

Processes

- faster reactions in making kivy features up to date with the technology
- More frequent release of bug patches.
- More frequent, smaller updates to show it is "alive"
- shorter release cycles
- issues clearing
- Need PRs being pushed back.

Code Quality

- Testing
- Cover the framework with more unit tests
- Automation of release processes
- type annotations
- Type annotations
- adoption of newer technologies
- Stability
- Fix import side effects
- Click-through testing similar to Selenium.

Development Tools

Hot Reloader

- Hot reload for speed development
- Hotreload on mobile like flutter
- Some sort of hot reload ability like flutter supports that enables viewing effect of small changes in GUI quickly.
- Hot reload
- Implement hot reload feature. (Hopefully better than kaki, it's a mess)
- Hot reloading (only provided by kivyMD if im not wrong)
- hot reload
- hot reloading
- Official hot reload on Android and iOS
- Official Hot/Live reload: There is a lack of an official, well-documented, and complete implementation to enable real-time debugging on the same device and across different devices (desktop <> mobile).
- HotReload for mobile or maybe with an emulator.

IDE Support (including WYSIWYG)

- Gui design tool like QT designer (I know, it would be a major project on its own ;-)
- visual UI designer
- Ui designer

- WYSIWYG rich text editor widget and display.
- IDE support for KVM files and code completion
- Ide Design
- Create an IDE plugin (PyCharm)
- Drag and drop facility for simple UI components like buttons and labels while designing the .kv UI in some dedicated editor.
- wysiwyg ui designer
- A WSIWIG designer program
- Drag and drop ui generation
- An LSP server for kv language
- Built in or 3rd party graphical editor, I need to see what it will look like on a a device without a recompilation
- Introduce Kivy studio so that people can drag and drop interfaces
- Figma to Kivy code tooling
- VS code Kv language tooling
- Kivy Studio for kv language
- Tools for debugging widget layouts and placement issues
- Make it easier to create ui

Project Templates

- A CLI tool for making basic setup and class creations
- Project architecture templates
- We need some project setup tooling
- project creation template
- also default recommended filepaths/project setup,

Android from Windows

- a simple way to compile an android executable for distribution (from windows)
- Buildozer must work in Windows.
- Build process is to hard and face issue on windows. (require windows build support)
- APK building in Windows (without WSL or virtualization)

Build Time Performance Improvements

- buildozer must be faster.
- Reduce apk build time when using buildozer
- Improve deployment speed on Android
- APK file size on Android could be smaller

Building and Packaging

- facilitate the packaging and installation of apps (this is difficult in python in general)
- A robust packaging and distribution...yes, Pyinstaller is there but we are all scared of its failures
- better packaging toolings for apps.
- Create a better packaging experience for all platforms in one-tap
- a plug and play repo with some GH actions that would do the whole packaging process
- Buildozer: Automatically install missing libraries.
- need to properly expose templates used in apk building.
- Expose all android related project files to buildozer; if possible expose the android project file itself
- The compilation process is pain when you start a new project but after the first time it's just smooth
- Better support for cross compilation out of the box
- PyInstaller works but is a pain
- Reliable Build System
- more automation for iOS deployment (many manual steps in Xcode)
- Better integration with android ios tool chains
- a default way to package to windows/macos (if pyinstaller is default, why is it such a pain and NOT DOCUMENTED that pyinstaller's sys.
- Too hard for a newbie to package a python application

DOCUMENTATION

Vague calls for improvement

- better documentation in source code and in website.
- better docs
- better tutorials
- Documentation
- Documentation
- Add video tutorials on documentation / user guides.
- Create a YouTube channel for Kivy with community tutorials made by experienced/outstanding users.
- features are hard to find.
- maybe continue to gather more kivy projects and link them
- Documentation
- Better documentation
- Quality content from maintainers on how to do stuff with kivy, for example compiling kivy app with p4a.
- build an app marketplace... show that a lot of commercial apps can be done...
- API Documentation with better Versioning
- need better tutorials/update the wiki,
- Make your docs a little bit user friendly
- interactive and much more comprehensive documentation
- Marketing
- needs examples and a lot of non-coding work to get to that state, such as work on the kivy.org wiki
- Easier learning path
- handhold new users/ first time coders
- just needs a bit of polish for users to see and pick up to use seriously. New user experience guides/a robust helper community
- make Buildozer easier to use
- somehow lowering entry bar for new devs
- making android development easier
- tooling around kivy
- there is a great lack of functionality "out of the box"
- Debugging must be simplificated.
- need DX improvement

Examples

Calls for bigger examples

- Propose more complete example to show typical templates
- There needs to be MORE public examples that show things that people want
- There also needs to be a full stack python example on the kivy website, deployed to win/mac/android/ios btw.
- Delete most of the examples and develop better/more relevant ones, focusing on complete applications.
- Can we get some prebuilt samples of applications in kivy examples

Calls for more examples

- Include more minimal reproducible examples.
- More examples.
- More examples using kvlang
- More examples.
- Need more examples and tutorials. I have a problem of figuring out how to do things such as file saving on Android.
- More tutorials and tips
- need MORE EXAMPLES,
- guides/examples in all languages for people to follow.

Other Documentation Requests

- Add more images to documentation
- Including dual syntax in examples: Python/kvlang.
- Document best practices for Python <> kvlang interaction.
- Documentation an more example usage of async
- Improved installation instructions ,there are many challenges while installing kivy on wsl windows subsystem for Linux.

- Tutorials for efficient code on the internet (YouTube) you find codes that work but are not very clean, they do not use the interesting tools kivy gives to us
- Images and videos on kivy documentation showing the usage of each of the properties of the widgets, real examples.
- Official up to date text and video tutorials on packaging kivy app to multiple platforms
- Official text and video tutorials on building apk for iOS
- improve the hello world tutorial, for all platforms.
- text and video tutorials for advanced usage of jnius
- Instruction to package apps for Mobile devices and PC.
- Documentation regarding "name" vs "id"
- Some extra component testing tutorials for junior developers.
- Explain how to publish kivy apps with python
- give info on how to make your own recipe with examples,
- Explain how to get working on VirtualBox (turn on 3d graphics acceleration on VirtualBox.)
- Better Documentation on Pagers and Bindings

KIVY APP FUNCTIONALITY

Themes

Themes should be customizable

- Better theming and theme options built-in.
- Modern widgets: This involves a complete redesign of widgets to allow for a higher level of customization and a more attractive visual, eliminating the need for additional libraries such as kivymd, etc, to achieve a more modern look.
- An easy way to swap the theme of all widgets at once without having to use custom widgets would be amazing
- Improve layouts theming
- Add themes, more variety/choices in widget appearance.
- Theme support
- KV graphics template
- More focus on customizability.
- Different skins for widgets
- Easy way for theming
- easier theming system (maybe something like css)
- Ability to change filechooser design in more detail.
- More documentation around how to make custom widget styles.

Themes should be provided

- some good modern themes, as kivy has its own look, it looks outdated compared to current host UI
- some UI themes to choose from
- Update the default style with some KivyMD stuff.
- Get rid of the old grey buttons and pick a Tailwind theme and stick with it. This would make it easier to translate Figma mockups to a Kivy UI.

Widgets (General)

Vague Requests for Cleanup

- Modern widgets
- modernization of UI
- upgrade design, like kivymd and akivymd did.
- a better gui framework integrated, like kivyMD
- More modern UI widgets.
- More features/widgets
- Default theme looks a bit dated
- a good looking UI

Widgets should not be based on fixed images

- Default widgets should not use border images.
- default images on widgets are ugly.

- Improvements in SmoothLine, BoxShadow, and the new antialiased graphics in Kivy 2.3.0 would enable widget design based solely on the primitives, eliminating the use of images for each component of a widget.

Native Widgets should be supported

- What I'm looking for is a more refined UI which is provided by native languages on windows but kivy is not able to achieve that
- Map
- Access to native shared dialogs (OpenFile, SaveFile, ...) on desktop
- use the native textinput of each platform

Selectability

- Add ""selectable"" in Label widget, like in Flet (https://flet.dev/docs/controls/text/#selectable);
- (Seemingly) simple features like selecting label text
- selecting is slow, and doesn't works well on android

Non-Latin Support

- Support Arabic text
- Non-Latin language support
- Add cross-platform RTL text support
- Text layout for the language that doesn't use spaces to separate words.
- Emoji supported label and text input
- Emojis
- access emojis as input

Vague requests for more widgets

- More widgets
- More basic widgets

Widgets (Specific)

Video Player

- Customizable kivy.uix.videoplayer by separating its control to user defined widgets
- kivy.uix.videoplayer customization
- Kivy video and videoplayer
- Video player that works on iOS.
- Performant video player
- support for video streaming (HLS)
- The Video widgets should be able to do whatever VideoPlayer widget is doing except showing the controls
- A working video player example.

RecycleView

- The RecycleView - a lot of work to get it functioning
- Being able to add widgets to RecycleView while maintaining the exact view (rather than scrolling away)

ScrollView

- Scrollview widget really needs an overhaul - it can be difficult to use other widgets that require more than just simple tap/clicks inside one (such as other scrollviews or sliders).
- It would be nice to have more options for scrollviews as well - 'traditional' scrollbars with arrows, ability to disable 'flick' scrolling, allow scrolling with only certain mouse buttons (right click, middle-mouse click)
- Maybe converting scrolling into a behavior and/or splitting scrollbars into a separate class could improve usability and functionality.
- Scrollview does not allow me to resize each widget individually only the whole one
- Fast view pictures in screen in scrollview
- ScrollView performance

PDF Viewer

- in-built pdf reader/renderer
- Implement PDFreader
- PDF viewer widget

Mobile Ads

- admob
- Kivy based mobile ads

Browser Widget

- A web viewer and a widget for a browser
- webview
- WebView for desktop
- An in app Web view / browser
- browser support.
- support embedded browser on desktop platforms
- I have difficulty opening the browser inside kivy. And this is a payment system that provides a secure channel for receiving information
- web viewer

Other New Widgets

- Plotters more beauty and with more functionalities and 3D models widgets
- Seamless screen switches
- Proper date picker
- most loved emoji and icons
- Font manager that display system font
- Add get_system_fonts() function that returns a list of system fonts

Other Widget Enhancements

- The on_release / on_press argument is missing in the arguments of the button constructor the ability to attach widgets to layout in the constructor For myself, I decided not to use "files.kvlang" and such problems bother me.
- When you set the picture with the background of the buttons, if the size does not match, it loses a lot in quality.
- a paging widget

Concurrency

- Add multithreading support
- Try adding your solutions to simplify the development of multithreaded applications, for example, the @another_thread decorator.
- Easy threading interface,
- Async
- Asynchronous usage for internal kivy implementations

- More friendly loading kv's on the enter of program with threads

Behaviors

- Improve API consistency. Widgets, layouts, and other elements do not always share common behaviors, concepts, and nomenclature. The point of improvement here would be, for example, if pos_hint works in a BoxLayout, it should work in a GridLayout; if padding accepts different values in one widget, it should accept in another, etc.
- Need improvement in some widgets and their behaviours
- Need to make kivy behaviours consistent
- Full and consistent implementation of basic state behaviors, such as HoverBehavior, FocusBehavior, ButtonBehavior, etc. (may include other convenient behavior types). Currently, there is no implementation of HoverBehavior.
- Theme loading system through a configuration file (such as .json or another specific format), similar to what happens with QT.
- Multithreading for the UI
- There are deficiencies in game development, for example image collision detection (collide_point) is not sensitive and runs slowly.
- Refactor all the gesture stuff to a proper system, need global gesture detection and composable, local detection.
- Widget animation. Tricks are required today to animate some property, and no solution really works well within KV.
- Widget mouseover events.
- drag and drop
- being able to play animations when a file is dragged over the window
- Fix MotionEvent dispatching/protocol - all the touch.ud[] stuff in ScrollView should be built on better abstractions.
- being able to play animations when a file is dragged over the window
- hover

KV Language

- There was a bug in the kv-parser where you cannot have a comment after a context instruction. This is a sharp edge for beginners.
- Some implementation of control structures in kvlang (if, for). Take advantage of other frameworks structures/way of doing things. Ex: VueJS control structures, django project architecture.
- kvlang looks a bit like CSS but offers much less flexibility, it could be made more like css
- Conditional widget rendering on KV language (for example, if some condition, the widget is not rendered)
- More intuitive connection between Python and KvLang code

Layouts

- Warnings or exceptions when setting a parameter that has no effect. (Qt has this) Labels or positioning is difficult to get right, and involves trial and error. You shouldn't be allowed to set the pos: parameter on a widget that is in a stacklayout, also you shouldn't be allowed to set the pos parameter if it is overruled by a pos_hint etc.
- The default values of widget attributes such as size hints seem not very useful and are tedious to override so often.
- Better handling of the widget placement in layouts which is often a bit of trial and error as the size/position hints don't always work as expected.
- Auto resizing widget by its children's size

Inputs

Virtual Keyboards

- The vKeyboard requires you to push shift while typing a letter like a physical keyboard. This behaviour makes no sense for touch.
- The virtual keyboard system is confusing and Kivy struggles with some softkeyboards on mobile platforms.
- Some manufacturer android vkeyboard dont works well out of the box with kivy.

Other Input Enhancements

- Adding functionality to support separate functions for stylus erase and select buttons (applicable for touchscreen devices with stylus support).
- Speech-to-text input
- Text input isn't perfect
- Kivy textinput, it has very weird behavior on android; hides words suggestions of keyboard, and other language's scripts

Back End Enhancements

Multi-window Support

- multi window support on desktop
- multi-window support for desktop apps.
- add multiwindow support

Web Back End

- kivy for web (pyodide)
- Support for web deployment
- web support for browsers.

Transparent Windows

- Transparent overlay support for raspberry pi 4
- transparent windows.

Hardware Acceleration

- ability to run accelerated on other platforms (eg rockpi)
- add cross-platform hardware-accelerated video rendering
- better accelerated graphics

Other Back End Support

- would be nice to allow some low level SDL coding [audio, video, inputs, stuff]
- kivy uses 3D with difficulty, but the result is obtained as from 2000, there is no normal 3D documentation
- Support for KMS/DRM overlays or competitive video playback on raspberry pi 4
- [Discussion on pros and cons of supporting WGPU]
- Support OpenGL ES 3.2 (making it visible?)
- gstreamer which sometimes just isn't available on some systems
- Add angle_sdl backend for macOS and iOS [Apple has deprecated OpenGL]
- Add the glGetProcAddress function so that developers can retrieve the necessary OpenGL functions themselves
- We can improve the supported graphics backend to include new age technologies like metal, vulkan, webgpu
- Textures functionalities to improve the images usage

Performance

Vague Performance Improvement Requests

- Speed
- Performance

- Android performance
- performance
- Speed.
- more performance improvements
- speed
- optimising the UI

- Performance for large widget number
- Performance
- Sometimes Kivy app appears dead.
- performance
- Higher frame rates

Specific Performance Improvements

- A tool to see/inspect performance bottleneck in complex UX.
- subsystem is very laggy, its ok for simple audio play, but cannot be used in time critical scenarios.

- fast widget creation; widgets are created too slowly in large numbers

Start Up Performance Improvements

- reduce app launch time
- App starting speed
- improving start time
- App startup time
- black screen shouldn't be there on startup
- Make LazyLoading of screens the default behavior (kulothunganug/kivy-lazy-loading-template)

- I think reducing the loading time of the application is the most important issue, users do not expect an application that does not open in 10 seconds, for example the bundel.so file is very large. By closing some packages from buildozer (some packages such as sqlite can be closed, but all non-essential packages should be closed), there is also the problem of loading sound files (when you try to load 6,7 sound files at the same time with SoundLoader, you have to wait 10 seconds), since I am developing games, game sounds are disabled. I need to install most of them at the same time.

Recipes

- Matplotlib is still not a first class citizen

- matplotlib

Deprecations

- I think at this point removing stuff is more important than adding features...
- Write libraries not frameworks. Python is such a rich ecosystem with a lot of tutorials on how to solve any problem. For example 'PYTHON' logging mode was a great addition for us. Maybe even things like the kivy settings or filechooser could be moved out of the core library.
- Fix config system - start with deleting ConfigProperty - in fact, delete everything related to ConfigParser, json panels, and kivy.uix.settings. Instead put an EventDispatcher with properties in App.config, and add some usable form layout tools.

- Declare the X11 window provider as deprecated. Major Linux distributions are actively transitioning to Wayland, and they may abandon X11 soon. Note that X11 support is present in SDL (https://wiki.libsdl.org/SDL2/Installation);
- Deprecate kivy.graphics.tesselator (https://kivy.org/doc/stable/api-kivy.graphics.tesselator.html)
- Remove kivy.garden and remove non-vital widgets from kivy.uix. Put the important/still usable stuff in kivy.contrib and take on the work to manage contributions and deprecate/remove/fix things that fall behind development.
- Remove all automatic loading of kv files.

Plyer-style Cross-Platform Abstractions

Permissions

- Communication with OS specific: Mostly permissions
- Make requesting permissions on android easier
- Newer permission problems.

Notifications

- Make it easy to package proper modern background notification system
- Push notifications
- dynamic notifications
- notification

Other

- Make it more compatible with android features like set app as default for specific file types
- Android API, can't interact with device components like camera or nfc feature
- Homogeneity between Linux and Android versions
- More advanced work with sound (For example, 3D sound)
- incorporation of new age hardware capabilities like biometrics, passkeys,
- Better integration with mobile platforms (notifications, sensors, camera record) with the emphasis of consistent support across android and iOS (so we can rely on the feature being present in both places)
- in some phones with android [some xiaomi] (audio?) inputs cannot be changed
- unified modules for different functionalities

ALREADY SUPPORTED REQUESTS?

- More robust toolchains for building / deploying across all platforms.
- Better iOS support for building / deploying iOS apps. Not sure if I can use buildozer yet for iOS apps? we have to do it via XCode still.
- Buildozer cannot work in a virtualenv.
- a tool of conversion to apk
- anti aliasing
- Anti Aliasing
- AsyncImage
- date and time picker
- Integration with kivymd / or a clone of it
- build to mobile (Android + iOS)

SCOPE EXPANSIONS

- Kivy is limited to python, so the knowledge and skills of kv lang can't be used with other popular programming languages.
- Allowing some sort of html and css to equivalent kv lang converter for simple UI components.

MISCELLANEOUS SUGGESTIONS

- We need to get closer to frameworks like Flutter
- forms (django style)audio
- When setting one thing, it affect other, like when setting the left, even not setting the center, it calcs that, so we can use animations with them.
- problems with buildozer when trying to use multiple files (I sometimes doubt that it works at all, although working .apks remind me that this is not the case).
- Fix issues that cause kivy console loading to fail. Even the kivy getting started guide apps are unable to load the kivy console.

- The clock feature becomes offset based on the program run time.
- Make metrics/DPI stuff dynamic at runtime.
- Fix the provider system - enumerate available devices in one system, and make the provider selection process controllable (so for example one Camera can explicitly specify to use OpenCV).
- default out of the box components like in kivymd.
- be able to configure the version of opengl
- Smoothness (I don't know why, kivy is just isn't smooth and robust in UI compared to flutter)
- Default UI,
- More integration with all platforms (for example robert-flatt's camera4kivy is just amazing and works well, if it in-buit in kivy, it would so cool as he also abandoned)
- Don't cache all kind of opengl/kv objects. We have to explicit unload kv objects to prevent weird "double" gui elements when switching kv objects in layouts.
- easier ability to create check boxes or drop downs from a list
- structure check boxes and check lists so that they can be iterated over to check state or trigger behavior
- need a similar workflow to React Native
- _MEIPASS puts all your stuff in a a temp folder. This is really not Kivy's fault but rather a Python ecosystem fault
- Weak-references leading to garbage collection is difficult for new people
- KivyMD disrespects Kivy's default setup.
- use poetry
- Support for developing 3D games
- easy incorporation of AI into our apps
- Add something like Platform.IS_WINDOWS_7_OR_HIGHER;
- kivy.graphics.Mesh - remove the limitation on 65535 indices (vertices) (https://kivy.org/doc/stable/api-kivy.graphics.html#kivy.graphics.Mesh).
- have better tooling around kivy from design to code and project setup
- we need screen setup and optimizations
- be able to design launcher app widgets
- adding more easy to use features for desktop app development
- navigation manager
- Import Adaptive widget error
- add general data models
- data persistence
- custom drawer where I can change extra round corners.
- Windows feature Like enhanced graphics, rounded screens
- introduce customizable icons