Vulkanised 2025

The 7th Vulkan Developer Conference
Cambridge, UK | February 11-13, 2025

vk-bootstrap: Vulkan Project Startup Made Easy

Charles Giessen
LunarG, Inc.

Who am I?

- Started learning Vulkan in 2017
- Joined LunarG in 2019
  - Maintain the Vulkan-Loader, Api dump, VkCube, Vulkaninfo, Vulkan-Utility-Libraries, SDK development, & more
- Joined the Vulkan Community Discord in 2018
  - Moderator since ~2021

Charles Giessen
LunarG

LUNARG®

GPU SOFTWARE SPECIALISTS

Has this ever happened to you?

- You get motivated to start a new Vulkan project
- You open up your editor
- You create an Instance
- You enumerate all Physical Devices
- You determine how to pick a Physical Device
- You create a Device, with the extensions you want to use
- You create a Swapchain
- You lose motivation
- Close your editor
- And remember why you don’t start new Vulkan projects

LUNARG

Introducing: vk-bootstrap

vk-bootstrap:

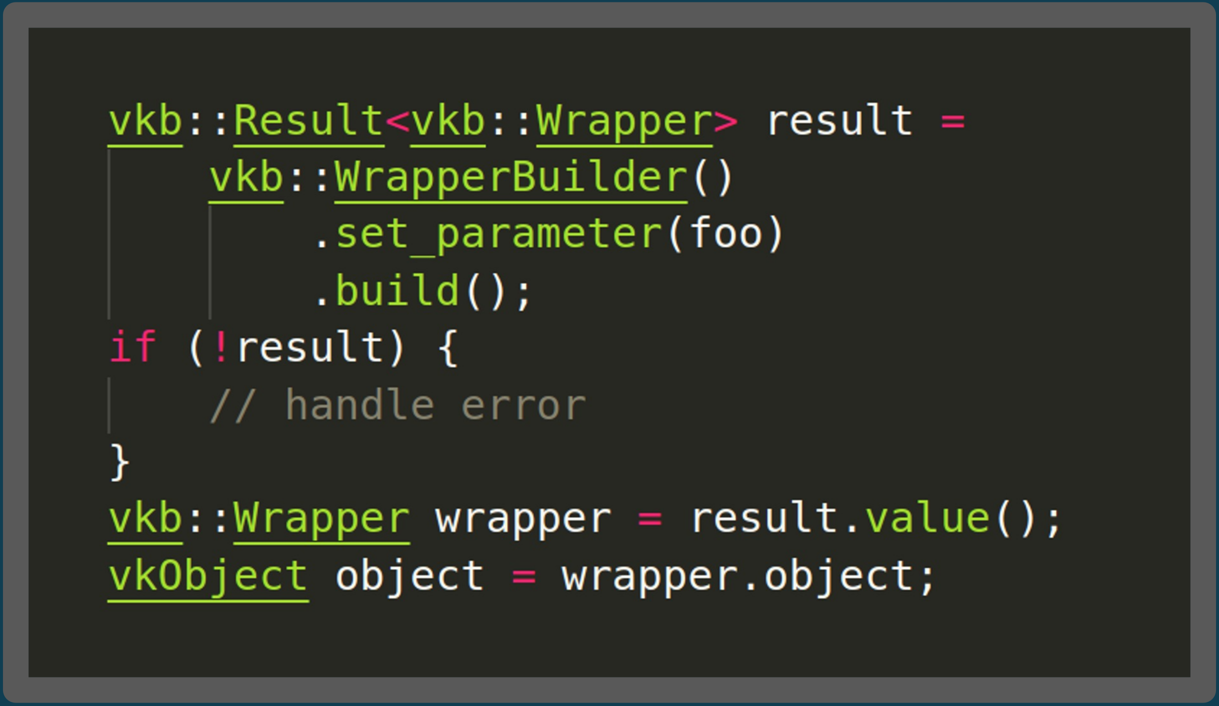
- Abstracts tedious Vulkan initialization boilerplate
  - Instance & Device
  - Physical Device selection
  - Swapchain creation & re-creation
- Builder Pattern API
  - Declarative and easy to use
- C++17 library with only a dependency on Vulkan-Headers
  - MIT License
- Easy to build:
  - Couple of header files and one source file
- Battle tested:
  - 4+ years old and used in countless projects

LUNARG

Let’s show some examples!

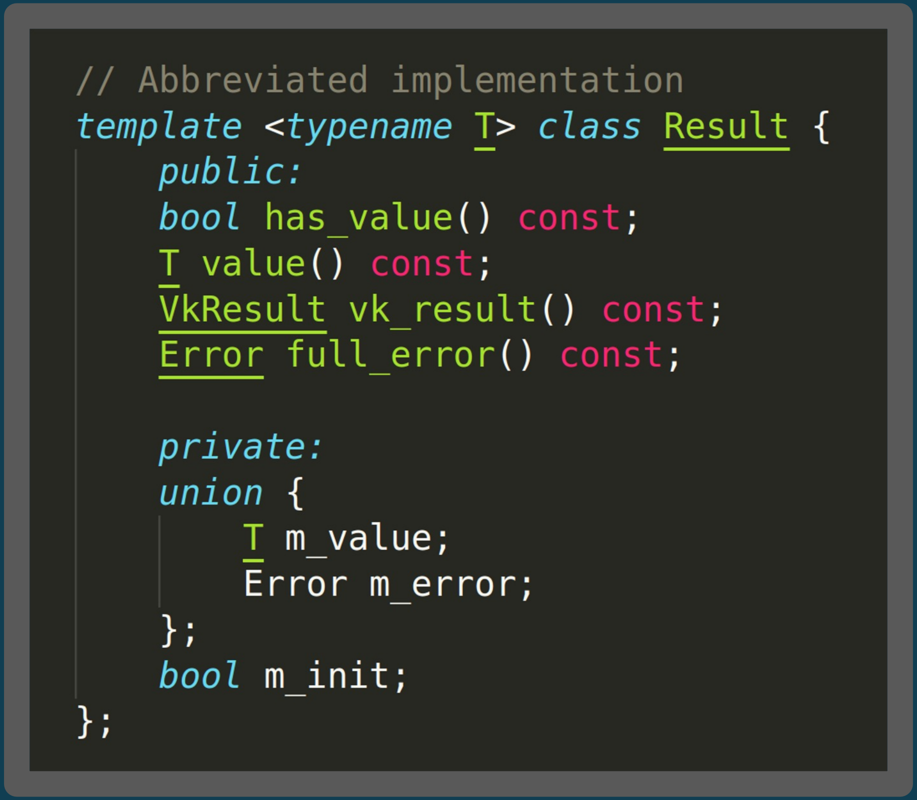
LUNARG

API Design



```
vkb::Result<vkb::Wrapper> result =
    vkb::WrapperBuilder()
        .set_parameter(foo)
        .build();
if (!result) {
    // handle error
}
vkb::Wrapper wrapper = result.value();
vkObject object = wrapper.object;
```

Custom Result Type



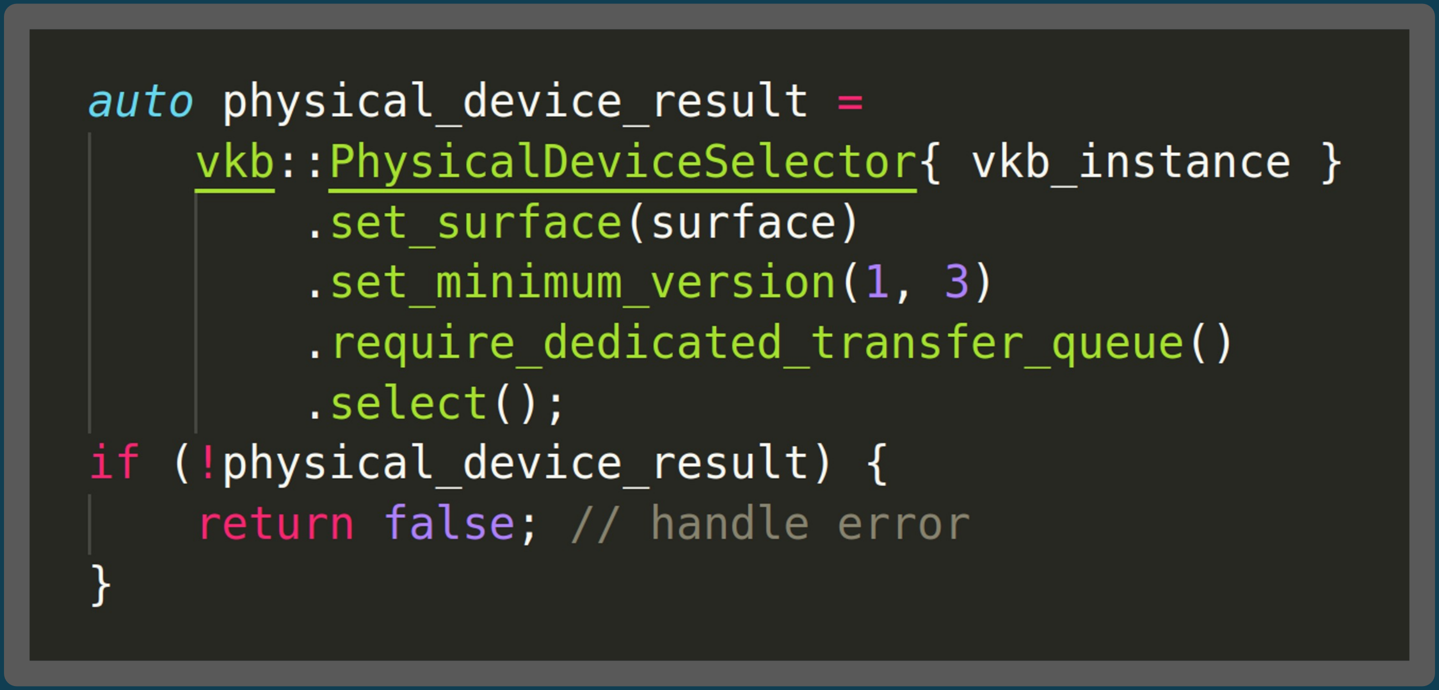
```
// Abbreviated implementation
template <typename T> class Result {
    public:
    bool has_value() const;
    T value() const;
    VkResult vk_result() const;
    Error full_error() const;

    private:
    union {
        T m_value;
        Error m_error;
    };
    bool m_init;
};
```

Instance Creation

```
auto instance_result =
    vkb::InstanceBuilder{}
        .set_app_name("Your Name Here")
        .request_validation_layers()
        .use_default_debug_messenger()
        .build();
if (!instance_result) {
    return false; // handle error
}
vkb::Instance vkb_instance = instance_result.value();
```

Physical Device Selection



```
auto physical_device_result =
    vkb::PhysicalDeviceSelector{ vkb_instance }
        .set_surface(surface)
        .set_minimum_version(1, 3)
        .require_dedicated_transfer_queue()
        .select();
if (!physical_device_result) {
    return false; // handle error
}
```

LUNARG

Select by hardware type

```
auto physical_device_result
    = vkb::PhysicalDeviceSelector{ vkb_instance }
        .prefer_gpu_device_type(vkb::PreferredDeviceType::integrated)
        .select();
if (!physical_device_result) {
    return false; // handle error
}
```

Requesting Core Features

```
auto physical_device_result =
    vkb::PhysicalDeviceSelector{ vkb_instance }
        .set_minimum_version(1, 3)
        .set_required_features_13({ .dynamicRendering = true })
        .select();
```

Requesting Extension Features

```
auto physical_device_result =
    vkb::PhysicalDeviceSelector{ vkb_instance }
        .add_required_extension(VK_KHR_DYNAMIC_RENDERING_EXTENSION_NAME)
        .add_required_extension_features(
            VkPhysicalDeviceDynamicRenderingFeaturesKHR{
                .dynamicRendering = true })
        .select();
```

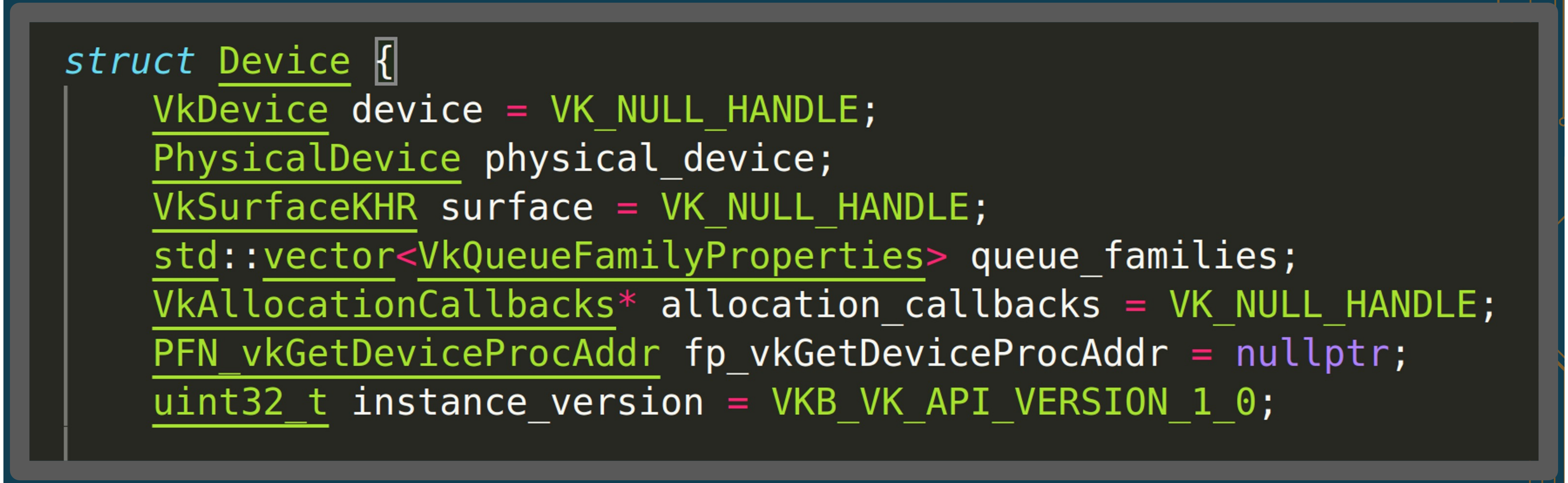
Device Creation

```
auto device_result =
    vkb::DeviceBuilder{ physical_device_result.value() }.build();
if (!device_result) {
    return false; // handle error
}

vkb::Device vkb_device = device_result.value();
```

LUNARG

Anatomy of vkb::Device



```
struct Device {
    VkDevice device = VK_NULL_HANDLE;
    PhysicalDevice physical_device;
    VkSurfaceKHR surface = VK_NULL_HANDLE;
    std::vector<VkQueueFamilyProperties> queue_families;
    VkAllocationCallbacks* allocation_callbacks = VK_NULL_HANDLE;
    PFN_vkGetDeviceProcAddr fp_vkGetDeviceProcAddr = nullptr;
    uint32_t instance_version = VKB_VK_API_VERSION_1_0;
```

Queue Retrieval

```
auto graphics_queue_result = vkb_device.get_queue(vkb::QueueType::graphics);
if (!graphics_queue_result) {
    return false; // handle error
}
VkQueue graphics_queue = graphics_queue_result.value();
```

LUNARG

Swapchain Creation

```cpp
auto swapchain_result =
    vkb::SwapchainBuilder{ vkb_device }
        .set_desired_present_mode(VK_PRESENT_MODE_FIFO_KHR)
        .set_desired_format(
            { VK_FORMAT_R8G8B8A8_SRGB, VK_COLOR_SPACE_SRGB_NONLINEAR_KHR })
        .set_old_swapchain(old_swapchain)
        .build();
if (!swapchain_result) {
    return false;
}
vkb::Swapchain vkb_swapchain = swapchain_result.value();
```

Get on with Vulkan!

- Grab the Vulkan handles from the Wrapper structures
  - vk-bootstrap is not a whole vulkan framework
- Go and write actually interesting Vulkan code

LUNARG

Cleanup

```
vkb::destroy_swapchain(vkb_swapchain);
vkb::destroy_device(vkb_device);
vkb::destroy_surface(vkb_instance, surface);
vkb::destroy_instance(vkb_instance);
```

Smattering of other Features

- Set custom Debug Callback
- Supports ‘headless’ contexts
- Get all Physical Devices which are suitable
  - Useful for lettings end users pick a Physical Device
- Supports all Physical Device Features structs
- Enable optional Device extensions & features
- Get a table of Device function pointers
  - For best performance

LUNARG

Integration

CMake Integration

```cmake
include(FetchContent)
FetchContent_Declare(
    vk-bootstrap
    GIT_REPOSITORY https://github.com/charles-lunarg/vk-bootstrap
    GIT_TAG        v1.4.307
)
FetchContent_MakeAvailable(vk-bootstrap)
...
target_link_libraries(YourProject vk-bootstrap::vk-bootstrap)
```

Fetch Content isn’t required

- git submodules if you prefer
- Available in vcpkg
- And available in conan

CMake not required either

- Just a couple of header files and single source file
- Easy to add into any Build System

Open Source Project

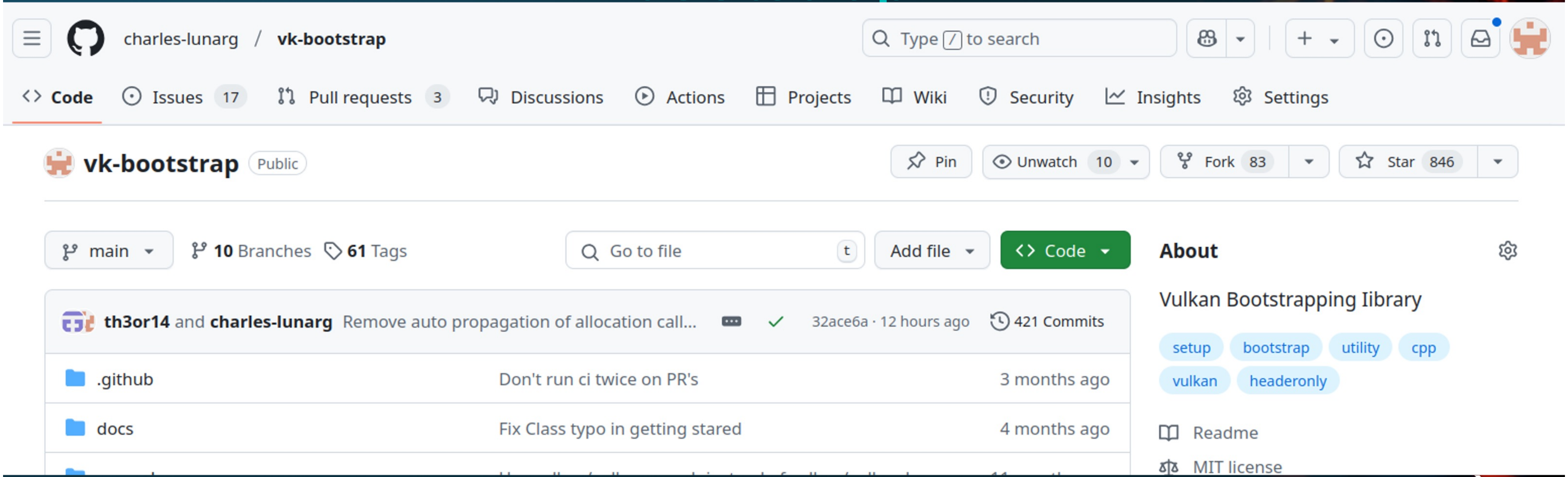
- Got tired of writing the same boilerplate code
- Created in early 2020
    - Design with “get it done then get out of the way”
    - Tried to make the right thing easy and wrong thing hard
- First working version in March of 2020
- Continuously updated over the years
    - Bug fixes and new features
- Many contributors
- HUGE Thank you everyone who’s contributed!

Future work

- Support Vulkan-Profiles
- Better Vulkan-HPP support
- Better VkQueue selection logic
- Your ideas here!

Where to find vk-bootstrap

https://github.com/charles-lunarg/vk-bootstrap/


charles-lunarg / vk-bootstrap

Code · Issues 17 · Pull requests 3 · Discussions · Actions · Projects · Wiki · Security · Insights · Settings

vk-bootstrap Public

Pin · Unwatch 10 · Fork 83 · Star 846

main · 10 Branches · 61 Tags · Go to file · Add file · Code

th3or14 and charles-lunarg Remove auto propagation of allocation call... 32ace6a · 12 hours ago · 421 Commits

.github	Don't run ci twice on PR's	3 months ago
docs	Fix Class typo in getting stared	4 months ago

About

Vulkan Bootstrapping Iibrary

setup · bootstrap · utility · cpp · vulkan · headeronly

Readme

MIT license

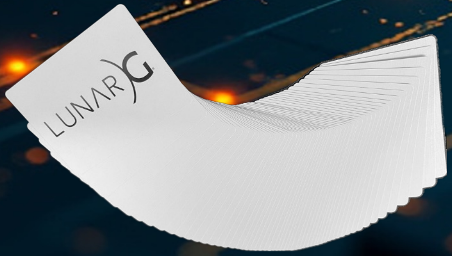
Thank you!

Actions

Download this Presentation

https://khr.io/1cr

Talk to us and get Swag!



Visit the
LunarG Sponsor Table

Take the Annual Developers Survey

https://khr.io/1cq

Your Feedback Matters!

Survey Results
➔ Are shared with the Khronos Vulkan Working Group
➔ Are used to drive development priorities throughout 2025

Survey Closes
Wednesday, Feb. 19, 2025
(GMT—7)