



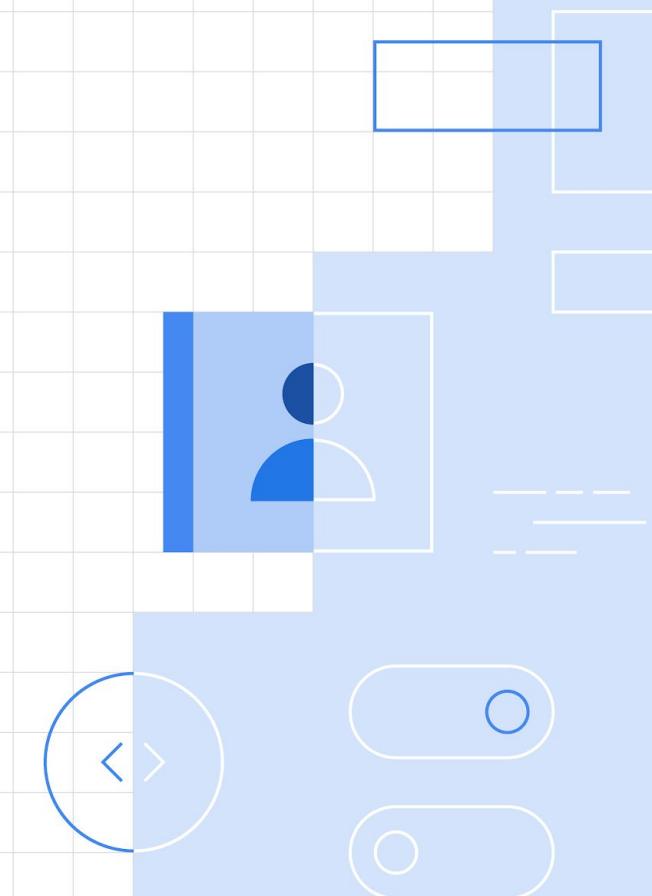
Using Gemini and Gemma to accelerate your Golang Application-building Experience

Saying 안녕하세요 to AI-charged workflows!



Suvaditya Mukherjee
ML GDE | GDG MAD
@halcyonrayes

Google Developers



\$whoami

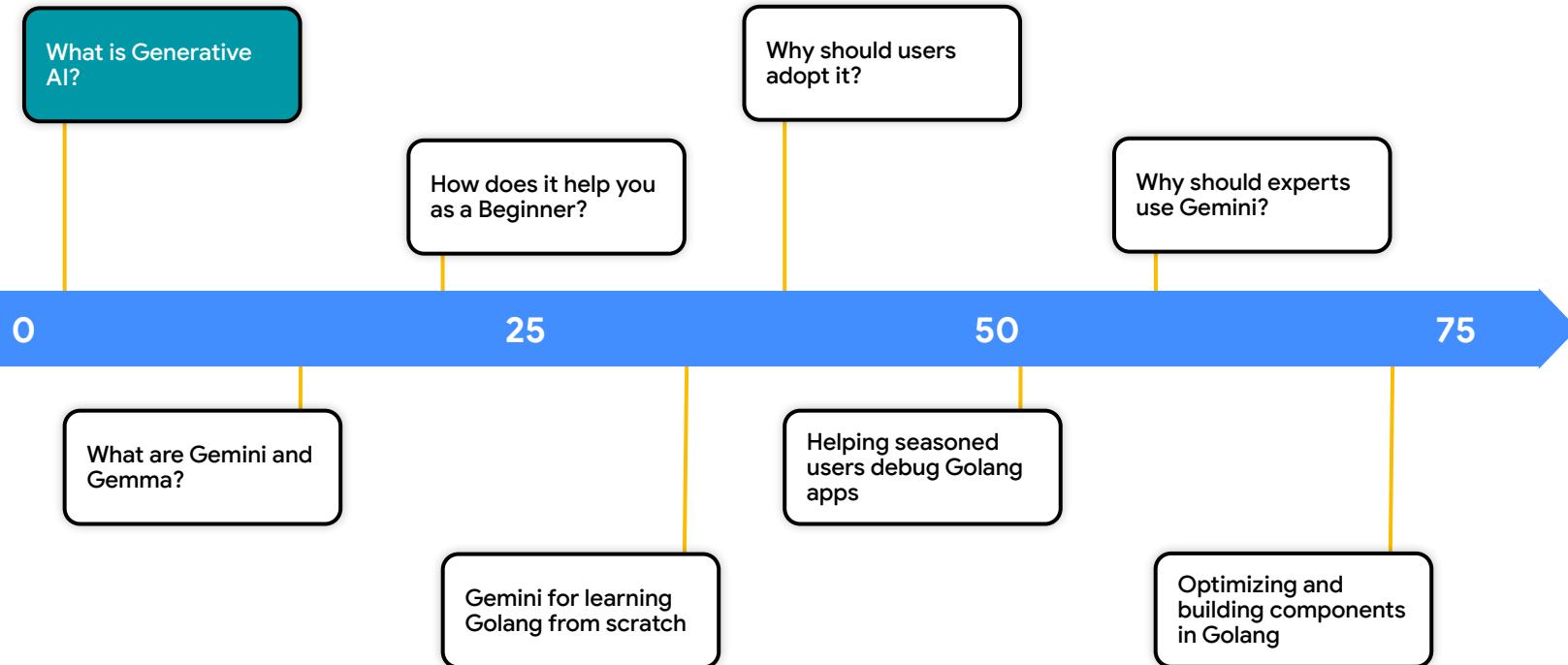
- Google Developer Expert (ML - TensorFlow Core/Keras)
- Master's Program in AI @ University of Southern California, Los Angeles, CA
- Ex ML Intern @ Harman International, IIIT-H, Ivy (YC W23), Mosaic Wellness
- Writing and running ML Experiments externally @ PyImageSearch
- Co-organizer @ GDG MAD, Mumbai, India
- GDSC Lead '21
- FC Barcelona supporter

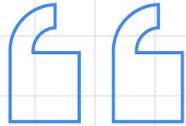
Some notes before we start

For the best experience. Literally.

- **Engage** during the talk, feel free to ask questions between sections but stick to the topic. Less of a talk, more of a conversation
- **Doubts are dangerous**, get them solved before we move further
- **I won't dive into too much math** unless needed. Feel free to grill me after the talk
- Slides and associated demo code will be made **available** right after the session
- After the session is complete, feel free to **ask off-topic questions** too

Things to cover today





**Machine learning is the ability of
computers to learn from data
without being explicitly
programmed.**

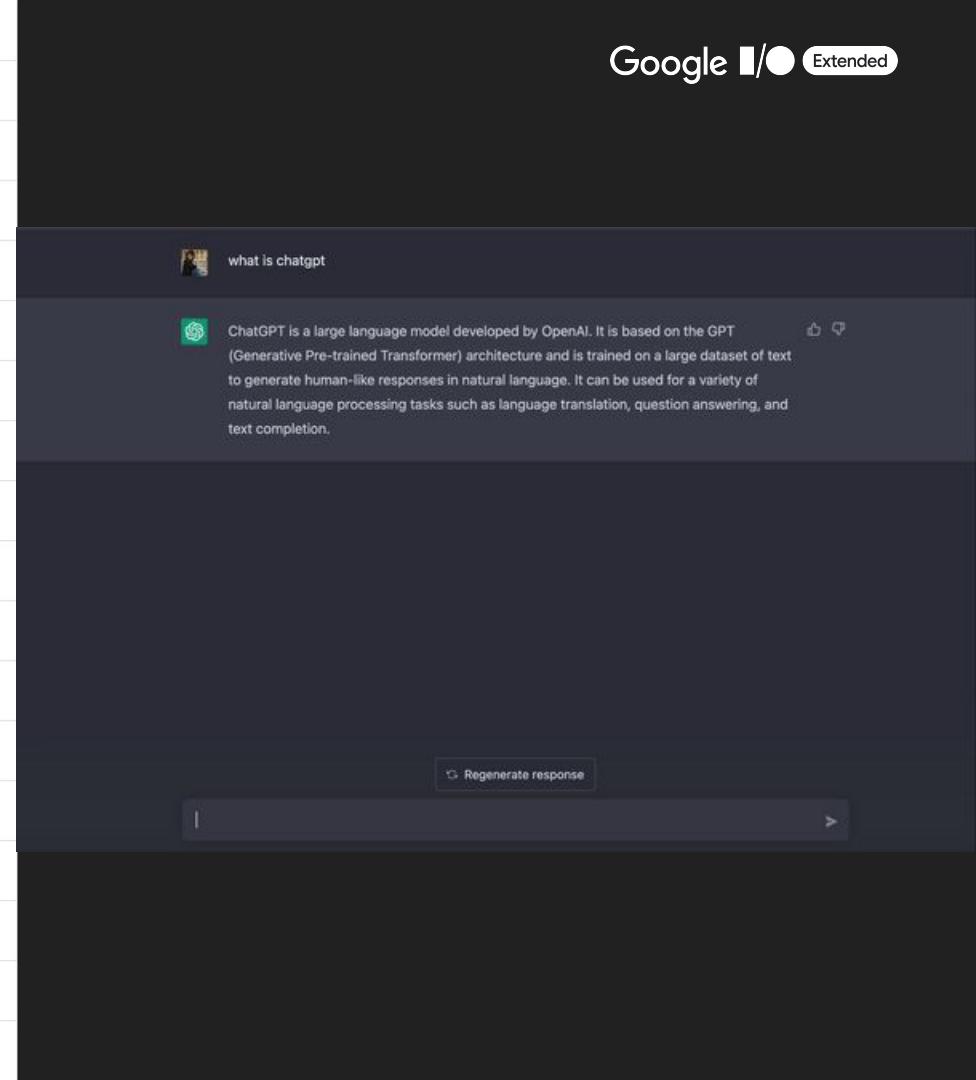
Gemini 1.5 Pro (a Neural Network, so it knows what it is talking about)



What is Generative AI?

It's literally in the name.

- Generate content of a certain modality like **Text, Image, Video, Audio**, etc.
- Different types like **LLMs** for Text and **LDMs** for Images





Of course you want to generate stuff like this too, no?

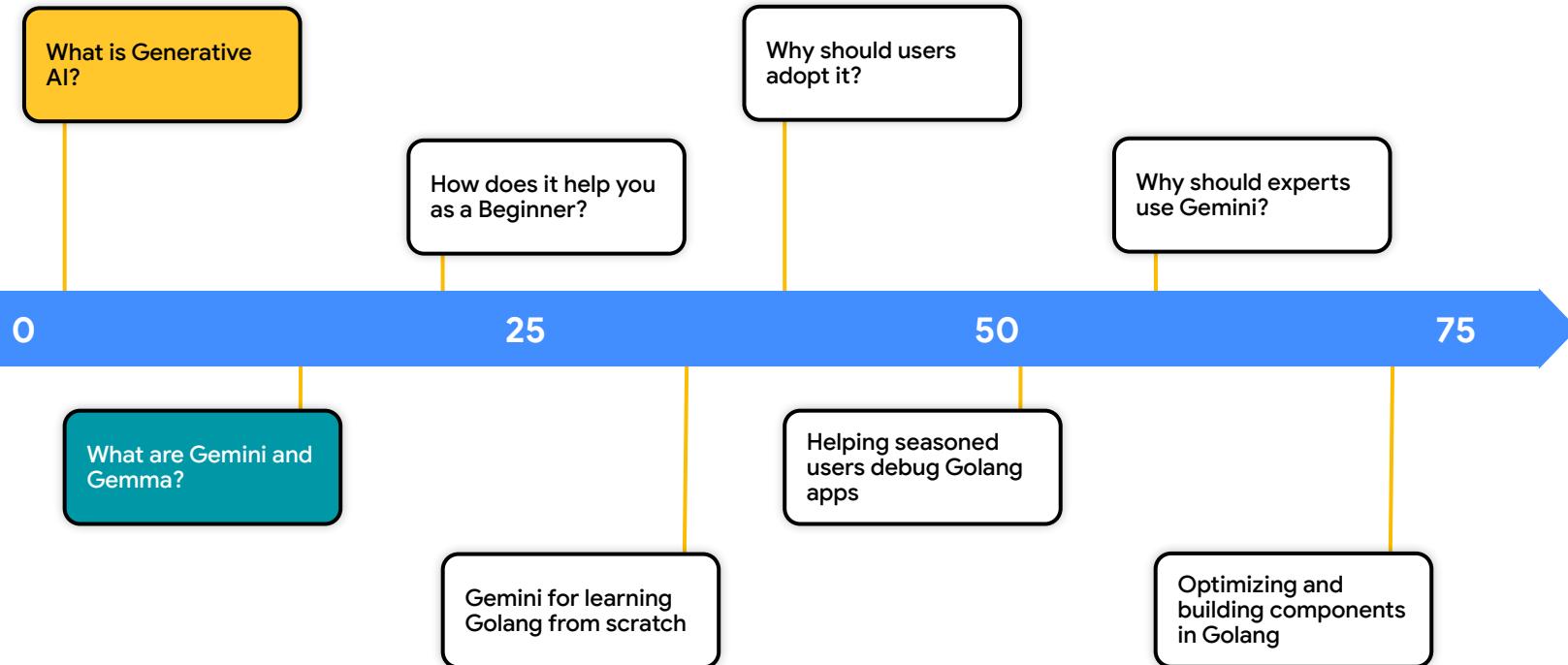
Good uses v/s Harmful uses

Same debate as data versus data (you read it differently, didn't you?)

- AI Risk or X-Risk
- Good uses
 - Business Chatbots,
Portrait Generation,
Summarizing information
- Bad uses
 - Fake videos (DeepFake),
Disinformation, and more



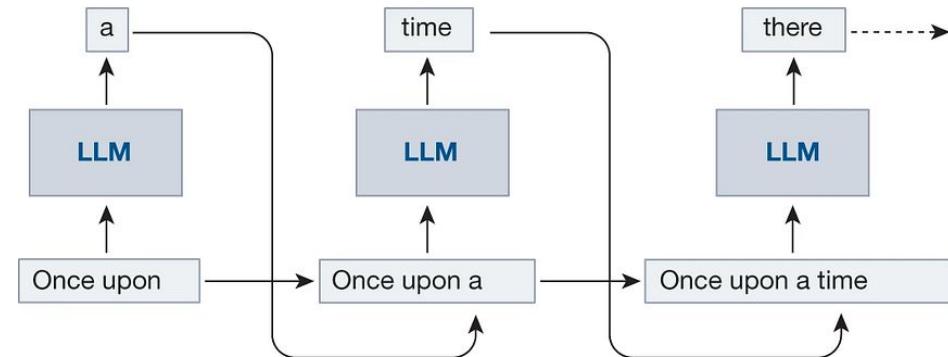
Things to cover today



Large Language Models

Algorithms to regurgitate text

- Use **Neural Networks** that mimic a simplified representation of neurons in a brain
- Learn from / “train” on **massive amounts of data** collected from the web
- Are capable enough to solve **different tasks across varied domains**



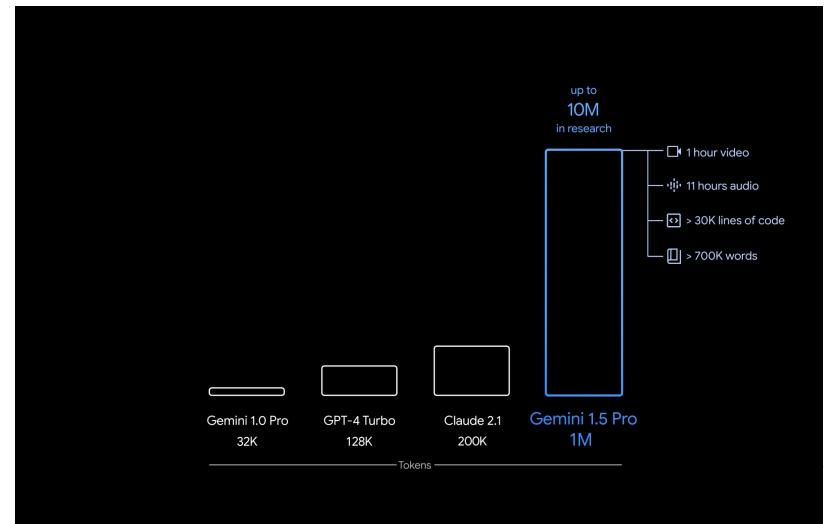
Gemini

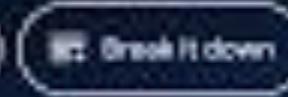


Gemini 1.5 Pro

A general problem-solving assistant

- **Largest context length ever** in a model
(effective upto **10 million tokens** as seen in research, public version has upto 1 million tokens)
- **Multi-modal** in nature (Can work with images, videos, and text)
- Has the ability to proficiently code in many languages like Python, C++, and **Go!**



Go    Break it down   Generate  Help me write  gle



Gemma

Gemma

A series of small and open LLMs for widespread use

Gemma Base

- 2B and 7B variants
- Trained on text, code, math

PaliGemma

- Gemma 2B and SigLIP merge
- Handles variety of vision-language tasks

CodeGemma

- Finetuned for Code Generation and Completion
- Can be run locally on small devices

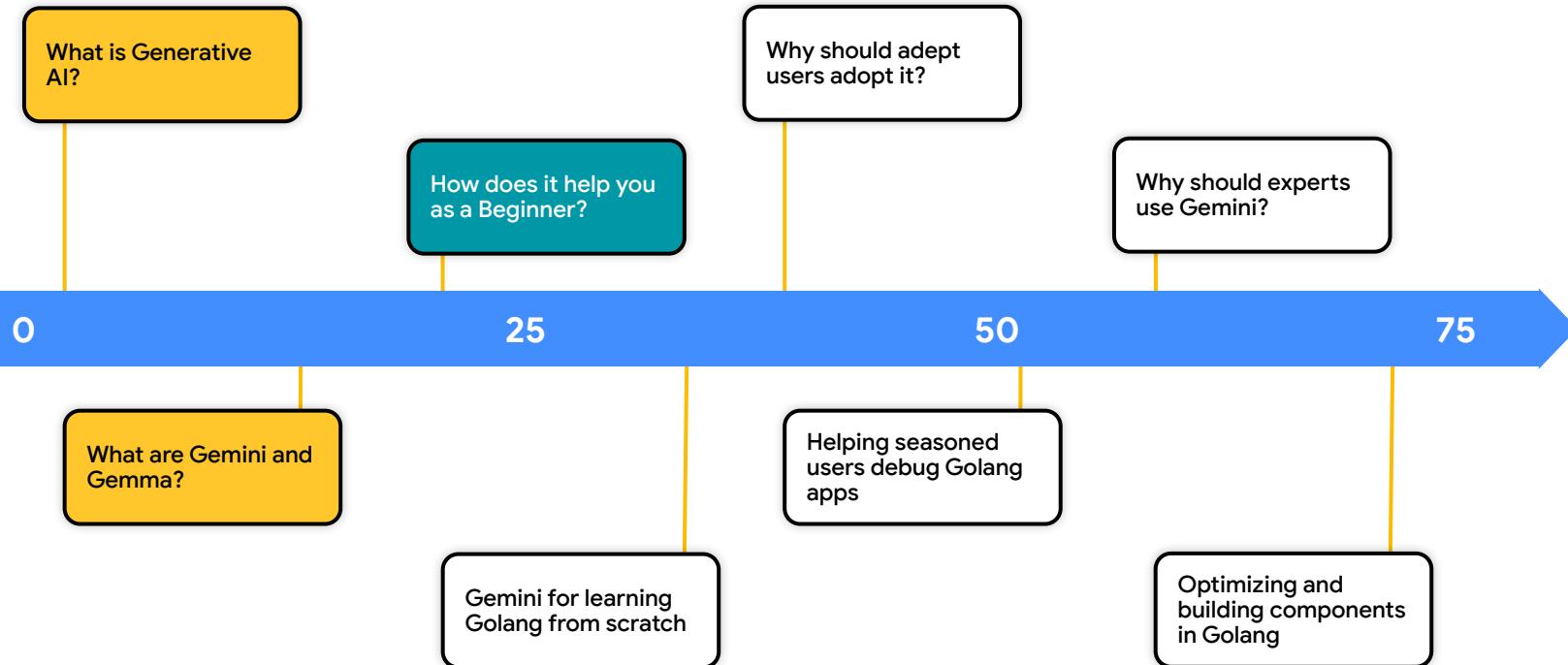
RecurrentGemma

- Memory-efficient
- Optimized for long-sequence tasks

A silhouette of a person sitting in a field, facing away from the camera towards a power tower in the distance under a sunset sky.

Google I/O

Things to cover today



\$44.89B

The market size for
Generative AI in Q1
2024



10x Productivity, 0x Cost

Generative AI for developer upskilling

01

Programming

02

Debugging

03

Creation

10x Productivity, 0x Cost

Generative AI for developer upskilling

01

Programming

10x Productivity, 0x Cost

Generative AI for developer upskilling

02

Debugging

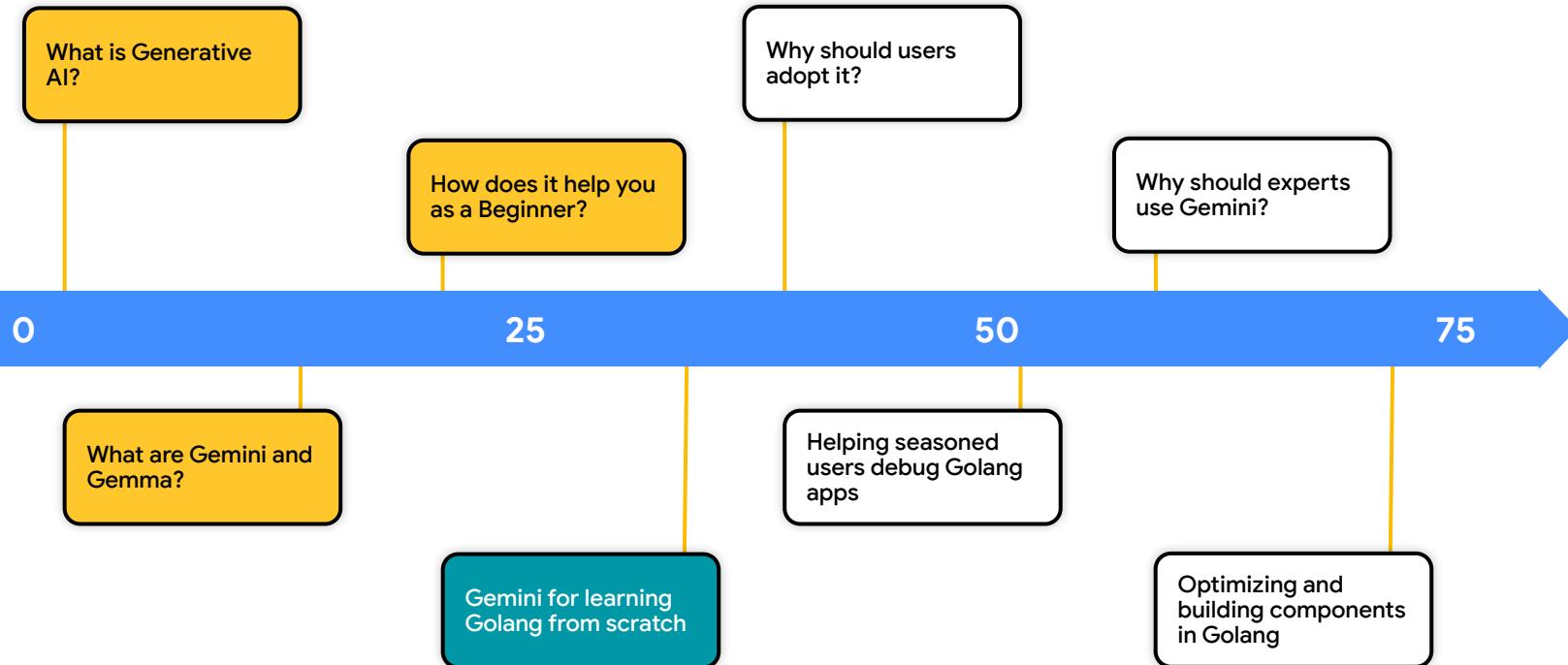
10x Productivity, 0x Cost

Generative AI for developer upskilling

03

Creation

Things to cover today



Experiment 1

Learning Golang as a beginner

Experiment 1

Learning Golang - Day 1

Hello, Suvaditya
How can I help you today?

Create a travel
itinerary for a city

Give me tips for how to
grow my YouTube
channel

Finish my gaming
podcast tagline: play,
win, and

Create a list of power
phrases for my resume



Experiment 1

Learning Golang - Day 1

 How can I learn Golang from scratch? Give me a game plan for doing so, important topics I need to focus on, and where I could learn them from.

Show drafts ▾ 🔍

◆ Absolutely, learning Golang can be a rewarding experience! Here's a roadmap to get you started:

1. Set up your environment:

- Install Go from the official website [[Golang download](#)].
- This will give you the necessary tools to run and write Go programs.

2. Grasp the fundamentals:

- **The Go Tour:** This interactive tour [[Golang Tour](#)] is an excellent place to begin. It introduces core concepts like variables, data types, functions, control flow, and more, all through a web interface where you can experiment with code snippets directly.
- **Variables & Data Types:** Understand how to store and manipulate data using variables and different data types like integers, strings, and booleans.
- **Control Flow:** Learn how to control the flow of your program using `if` statements, `for` loops, and other constructs.
- **Functions:** Define reusable blocks of code with functions to organize your program logic.

Experiment 1

Learning Golang - Day 1

3. Deepen your understanding:

- **Packages & Standard Library:** Explore Go's built-in packages like `fmt` for input/output and `math` for mathematical operations.
- **Pointers:** Pointers are a crucial concept in Go for memory management and working with addresses.
- **Arrays & Slices:** Learn how to store collections of data using arrays and slices, which are dynamic arrays.
- **Maps & Structures:** Organize data using maps (key-value pairs) and structures (custom data types).

4. Resources:

- **Official Documentation:** The official Go documentation [Golang documentation] is comprehensive and serves as a valuable reference.
- **Books:** Consider books like "The Go Programming Language" by Alan A. Donovan and Brian Kernighan for a structured learning path.
- **Online Courses:** Platforms like Coursera, edX, and Udemy offer various Go courses for beginners.
- **Codecademy:** For a more interactive approach, Codecademy's "Learn Go" course provides a hands-on introduction [Codecademy Learn Go].

Experiment 1

Learning Golang - Day 1

5. Practice & Projects:

- **Gophercises:** After grasping the basics, solidify your learning by tackling practical coding exercises on [Gophercises].
- **Start small projects:** Build simple command-line tools or web applications to apply your newfound knowledge.

Remember: Consistency is key. Dedicate some time daily or weekly to practice, and don't hesitate to seek help from online communities like the Go forum [Golang forum] if you get stuck.

By following these steps and leveraging the available resources, you'll be well on your way to mastering Golang!

Experiment 1

Learning Golang - Question 1



Can you design some exercises in the form of questions and possible solutions that I can practice Go with?

Show drafts ▾ 🔊



Go Exercises:

Here are some exercises categorized by difficulty level to help you practice Golang:

Beginner:

1. **Question:** Write a program that prints "Hello, World!" to the console.

Solution:

Learning Golang - Solution 1

```
package main

import "fmt"

func main() {
    fmt.Println("Hello, World!")
}
```

Learning Golang - Question 2

2. **Question:** Write a program that takes two numbers as input from the user and prints their sum.

Solution:

Learning Golang - Solution 2

```
package main

import (
    "fmt"
    "strconv"
)

func main() {
    var num1, num2 int
    fmt.Println("Enter two numbers:")
    fmt.Scanf("%d %d", &num1, &num2)
    sum := num1 + num2
    fmt.Printf("The sum is: %d\n", sum)
}
```

Learning Golang - Question 3

Intermediate:

- Question:** Write a program that checks if a number is even or odd.

Learning Golang - Solution 3

```
package main

import "fmt"

func main() {
    var num int
    fmt.Println("Enter a number:")
    fmt.Scanf("%d", &num)
    if num%2 == 0 {
        fmt.Println("The number is even.")
    } else {
        fmt.Println("The number is odd.")
    }
}
```

Learning Golang - Question 4

2. Question: Write a program that finds the factorial of a number using a loop.

Learning Golang - Solution 4

```
package main

import (
    "fmt"
)

func main() {
    var num int
    fmt.Println("Enter a number:")
    fmt.Scanf("%d", &num)
    factorial := 1
    for i := 1; i <= num; i++ {
        factorial *= i
    }
    fmt.Printf("The factorial of %d is %d\n", num, factorial)
}
```

Learning Golang - Question 5

Advanced:

- Question:** Write a program that defines a function to reverse a string.

Learning Golang - Solution 5

```
package main

import (
    "fmt"
)

func reverseString(str string) string {
    runes := []rune(str)
    for i, j := 0, len(runes)-1; i < j; i, j = i+1, j-1 {
        runes[i], runes[j] = runes[j], runes[i]
    }
    return string(runes)
}

func main() {
    var str string
    fmt.Println("Enter a string:")
    fmt.Scanf("%s", &str)
    reversed := reverseString(str)
    fmt.Printf("The reversed string is: %s\n", reversed)
}
```

Learning Golang - Question 6

2. **Question:** Write a program that uses a goroutine to print a message concurrently with the main program.

Learning Golang - Solution 6

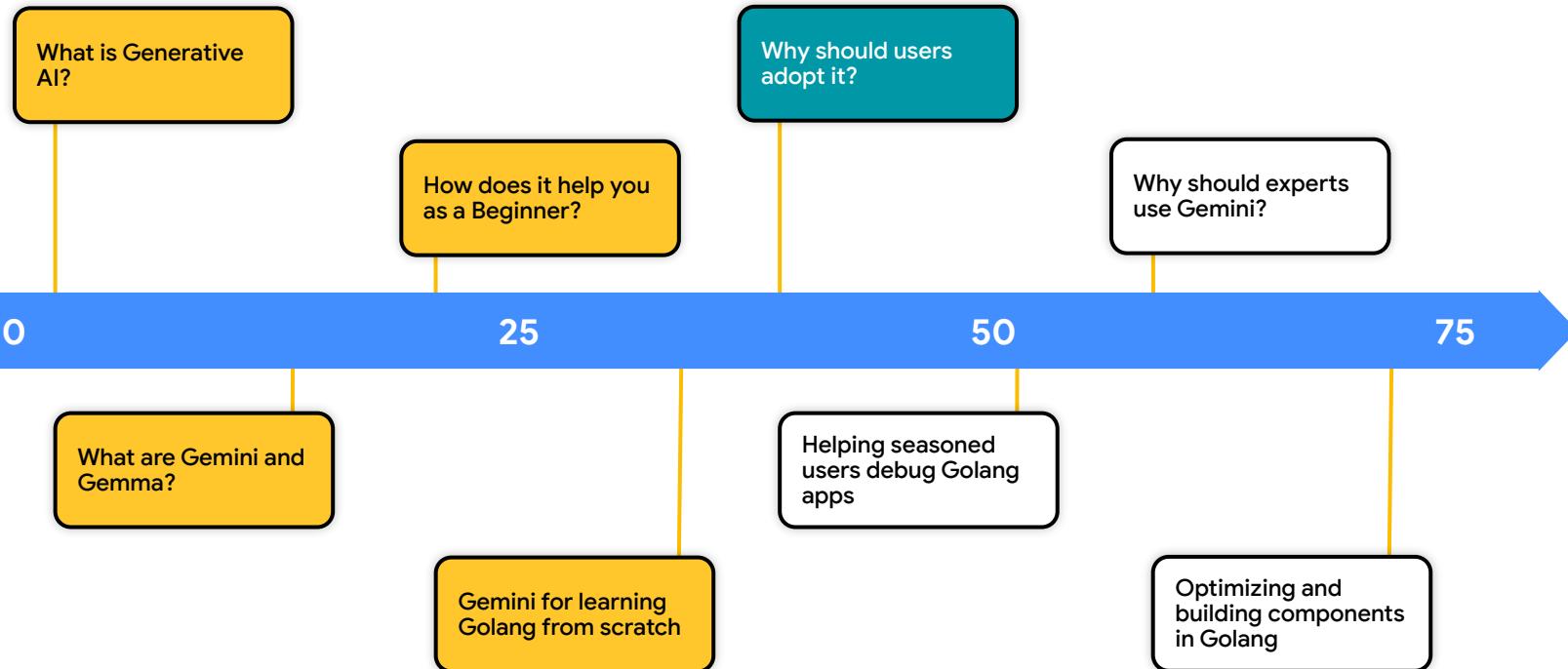
```
package main

import (
    "fmt"
    "time"
)

func printMessage() {
    fmt.Println("Concurrent message!")
}

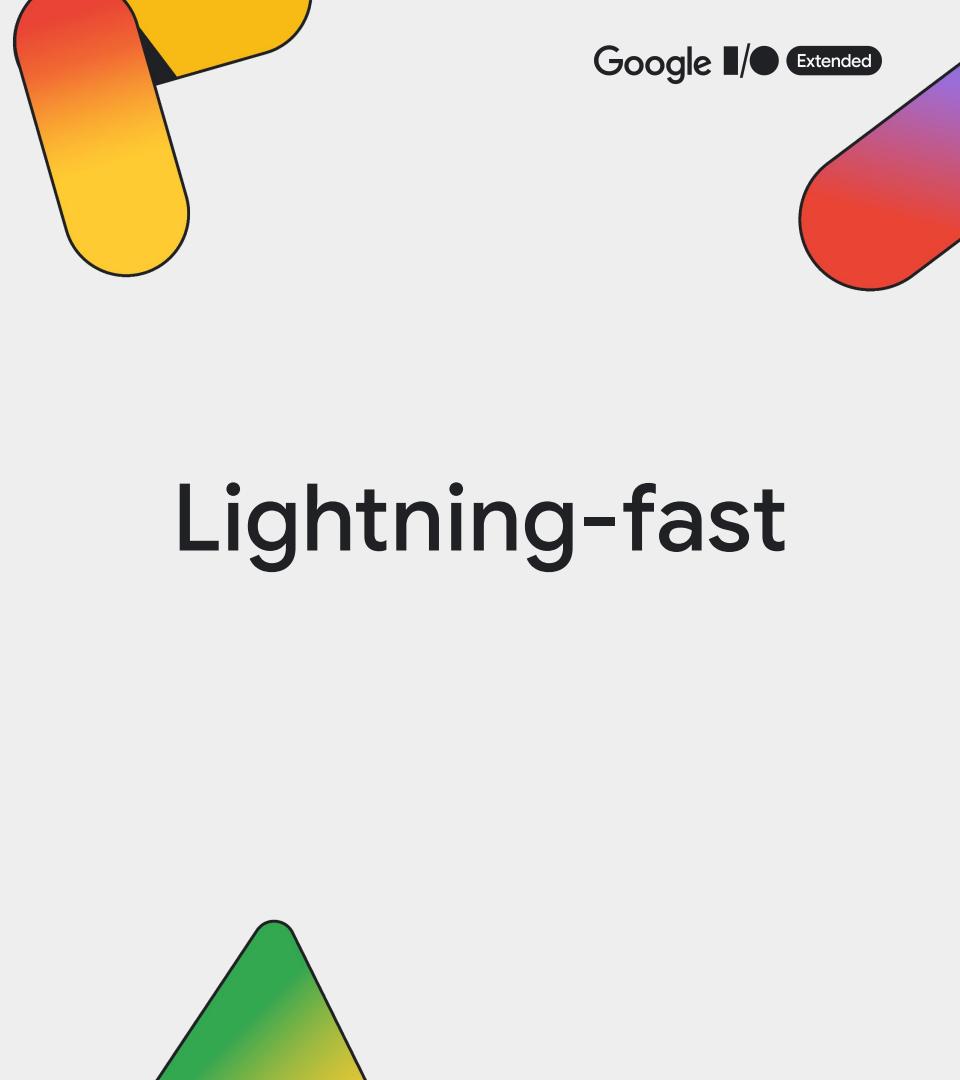
func main() {
    go printMessage()
    time.Sleep(1 * time.Second) // Wait for goroutine to finish
    fmt.Println("Main program finished.")
}
```

Things to cover today





Why should
users adopt it?



Lightning-fast

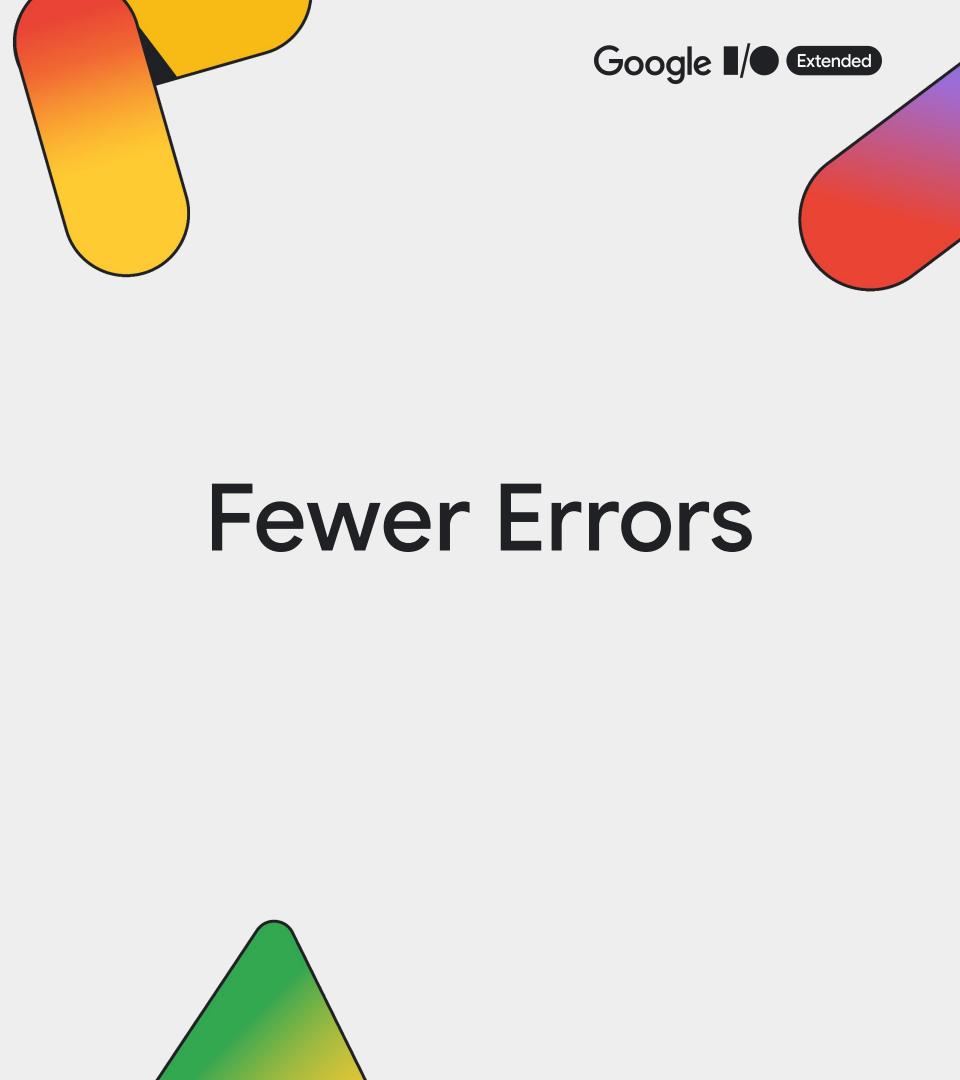


Why should
users adopt it?

Easy
Onboarding



Why should
users adopt it?



Fewer Errors



Why should
users adopt it?

Adapt to organization
practices



Why should
users adopt it?

Centralized
Knowledge

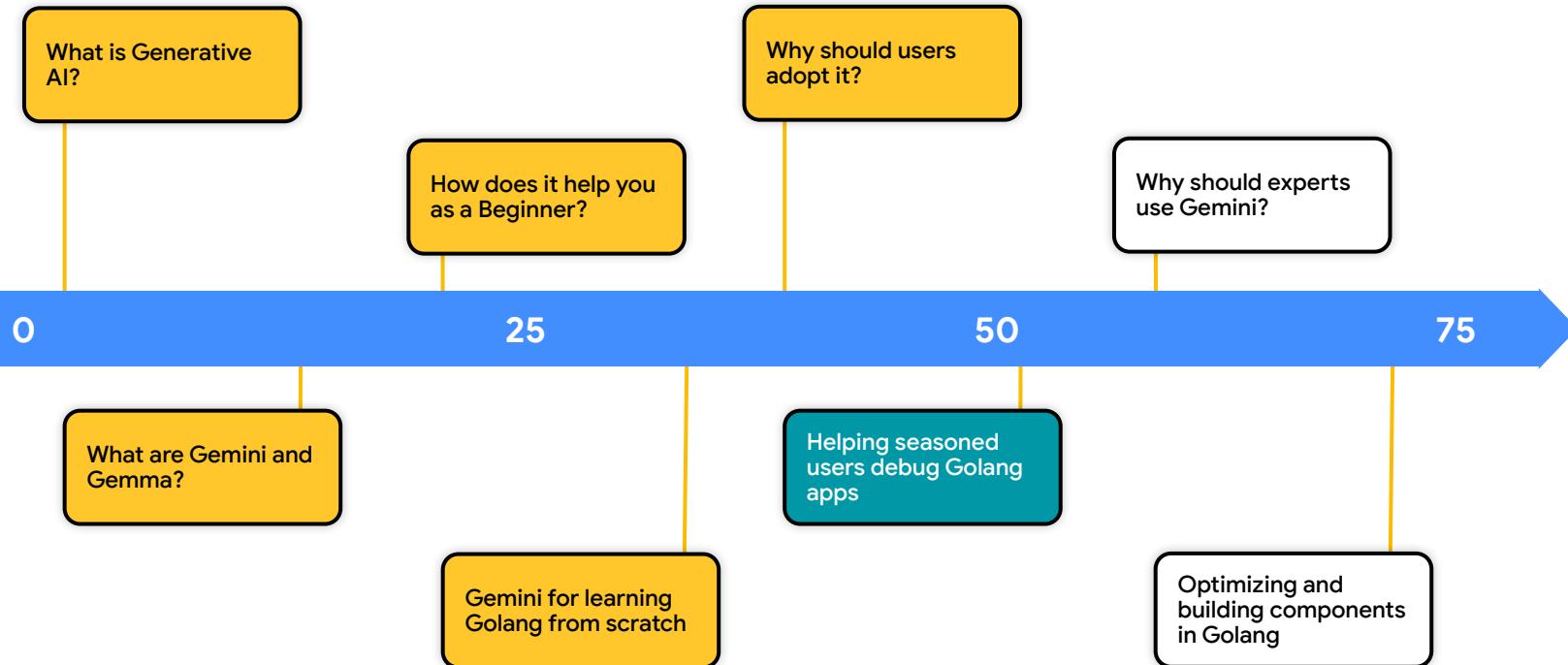


Why should
users adopt it?



Fast Prototyping

Things to cover today



Experiment 2

Debugging silent issues using Gemini

The Scenario

Our Golang server application **manages user sessions with a global variable**. We're experiencing **occasional data inconsistencies** and suspect a **race condition** might be involved.

Faulty code?

```
package main

import (
    "fmt"
)

var userSessions map[string]string // Global map to store
user sessions

func handleLogin(username, sessionID string) {
    userSessions[username] = sessionID // Update global
map
}

func handleLogout(username string) {
    delete(userSessions, username) // Delete from global
map
}
```

```
func main() {
    userSessions = make(map[string]string)

    go func() {
        // Simulate concurrent login requests
        for i := 0; i < 100; i++ {
            handleLogin(fmt.Sprintf("user%d", i),
            fmt.Sprintf("session_%d", i))
        }
    }()

    go func() {
        // Simulate concurrent logout requests
        for i := 0; i < 50; i++ {
            handleLogout(fmt.Sprintf("user%d", i))
        }
    }()

    // Wait for goroutines to finish
    fmt.Println("Finished simulating requests")
}
```

What does the code do?

- The code uses a global `userSessions` map to store user sessions (username and session ID pairs).
- The `handleLogin` and `handleLogout` functions directly modify this global map.
- We have two goroutines running concurrently: one simulating login requests and another simulating logout requests.

Experiment 2

Solution?

Google I/O Extended

Gemini

Throw it into Gemini!

Problem Solving 101

Going beyond learning



Here is some Go code. Identify the issue, explain it to me, and give me the corrected code with a solution to the error ▼

Experiment 2

Problem Solving 101

Going beyond learning



The issue with the code lies in concurrent access to the `userSessions` map. Go routines can run concurrently, meaning operations on the map by `handleLogin` and `handleLogout` might happen simultaneously. This can lead to race conditions.

Explanation:

1. **Race Condition:** Imagine `handleLogin` is adding a user-session pair for "user10" to the map. At the same time, `handleLogout` might be trying to delete the entry for "user10" (which might not even exist yet). This could result in unexpected behavior, like the user not being logged in even after a successful login attempt.
2. **Global Mutable State:** The `userSessions` map is a global variable, making it accessible by all goroutines. This creates a shared mutable state, a common source of concurrency issues in Go.

Solution:

We can fix this by using a mutex to synchronize access to the `userSessions` map. A mutex (mutual exclusion) ensures only one goroutine can access the map at a time, preventing race conditions.

Corrected Code!

```
package main

import (
    "fmt"
    "sync"
)

var userSessions map[string]string
var mutex sync.Mutex // Mutex for synchronization

func handleLogin(username, sessionID string) {
    mutex.Lock()
    defer mutex.Unlock()
    userSessions[username] = sessionID
}

func handleLogout(username string) {
    mutex.Lock()
    defer mutex.Unlock()
    delete(userSessions, username)
}
```

```
func main() {
    userSessions = make(map[string]string)
    mutex = sync.Mutex{}

    go func() {
        // Simulate concurrent login requests
        for i := 0; i < 100; i++ {
            handleLogin(fmt.Sprintf("user%d", i),
            fmt.Sprintf("session_%d", i))
        }
    }()

    go func() {
        // Simulate concurrent logout requests
        for i := 0; i < 50; i++ {
            handleLogout(fmt.Sprintf("user%d", i))
        }
    }()

    // Wait for goroutines to finish
    fmt.Println("Finished simulating requests")
}
```

Experiment 2

What changed?

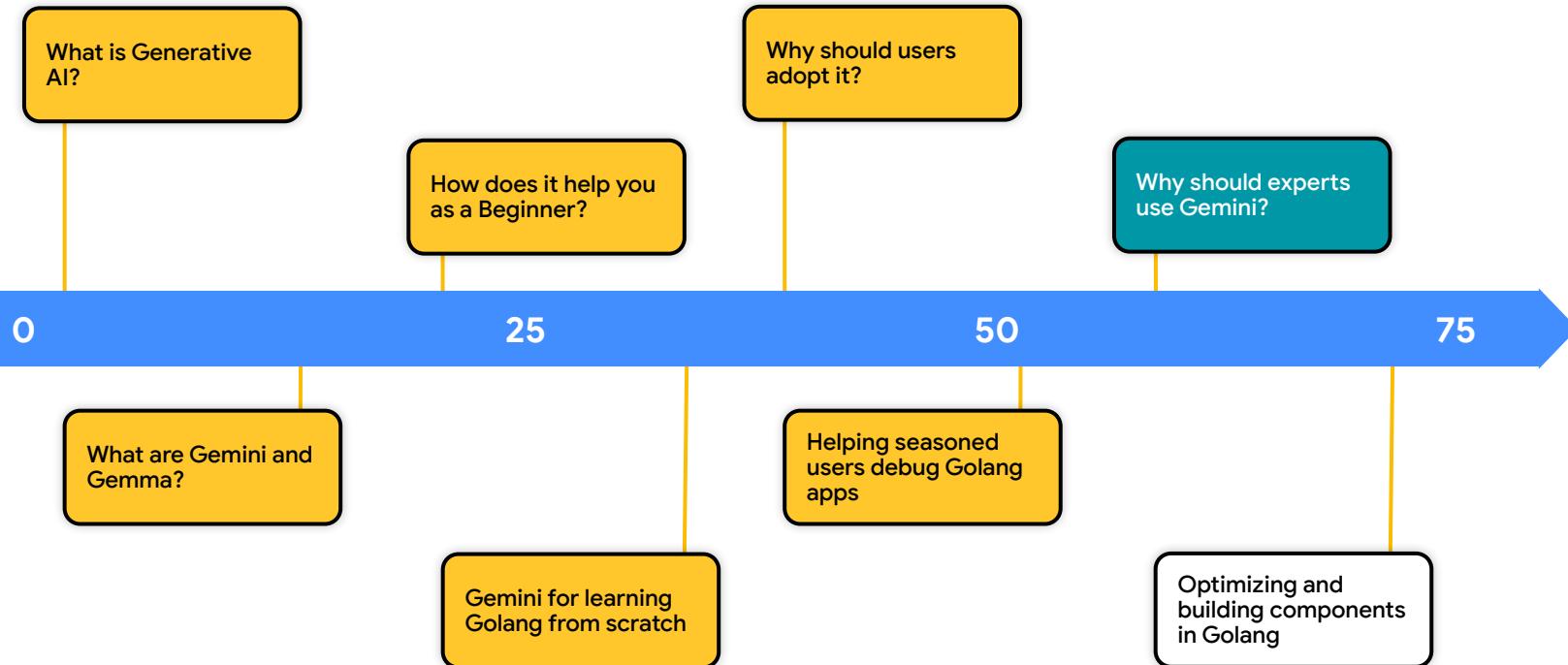


Explanation of changes:

1. We added a `sync.Mutex` variable named `mutex`.
2. Both `handleLogin` and `handleLogout` functions now acquire the mutex lock before accessing the `userSessions` map.
3. The `defer` statement ensures the mutex is unlocked even if there's an error during the operation. This prevents deadlocks.

With this change, the goroutines will access the `userSessions` map in a synchronized manner, eliminating the race condition and ensuring data consistency.

Things to cover today

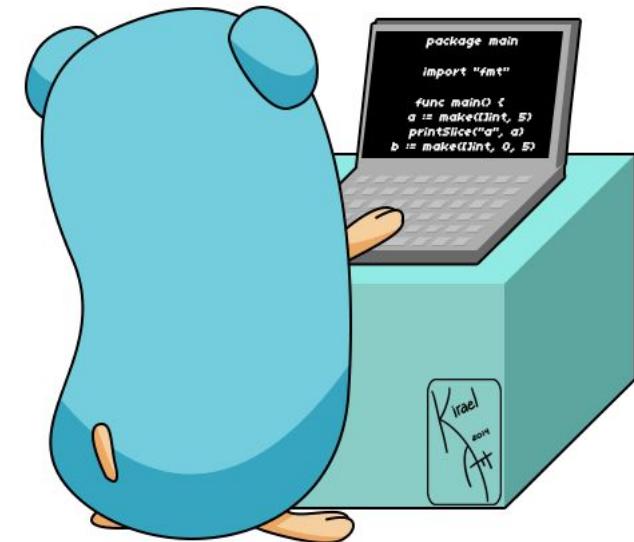


Gemini for Experts

Secret Sauce.

Performance Optimization

- Analyzing code and suggesting potential bottlenecks or areas for optimization.
- Identifying patterns in performance data and suggesting solutions based on best practices.

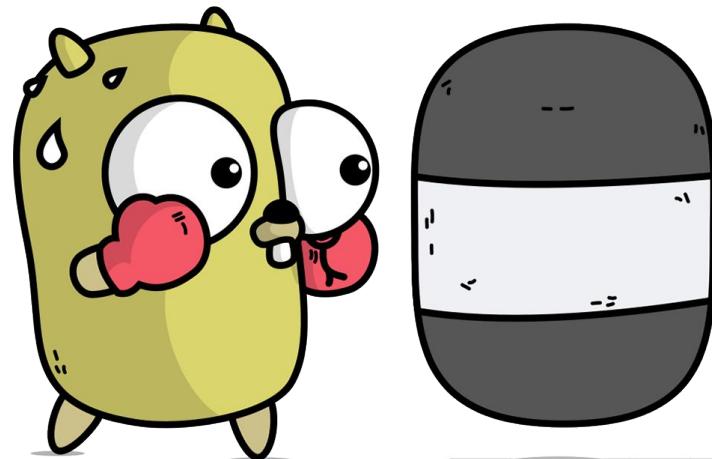


Gemini for Experts

Secret Sauce.

Security Auditing and Threat Detection

- Review Golang code to identify potential security vulnerabilities like common injection attacks or memory leaks.
- Analyze code for adherence to secure coding practices.
- Stay updated on emerging security threats relevant to Golang development.



Gemini for Experts

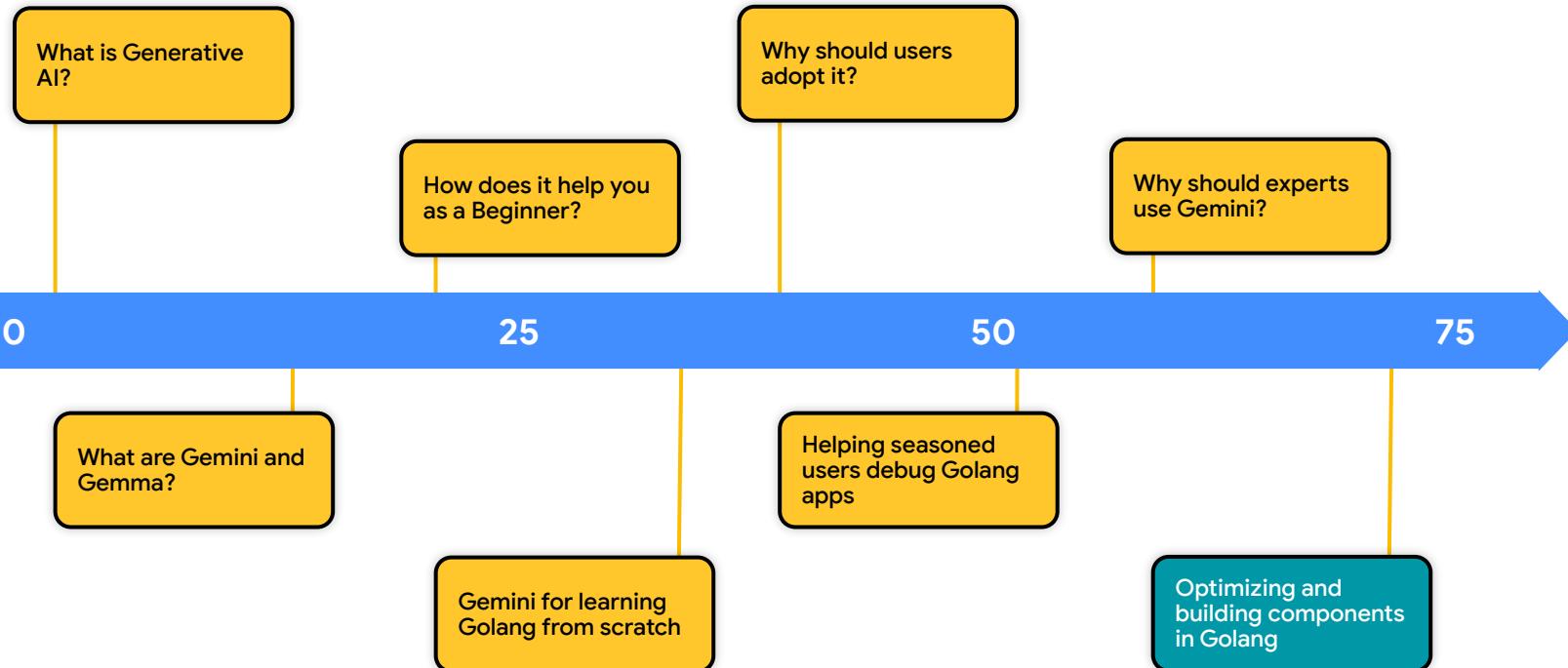
Secret Sauce.

Rapid Prototyping and Brainstorming

- Generating different user interface (UI) layouts and functionalities based on your descriptions.
- Brainstorming potential API designs and functionalities.
- Suggesting test cases based on your code's purpose.



Things to cover today



Experiment 3

Creating new applications from scratch using Gemini

Experiment 3

The Scenario

A Golang library that allows users to deploy and use [Python](#) functions

Experiment 3

What do we tell Gemini?



Write a golang library that helps general users to deploy python functions through Go. Write each file, and a small explanation of how it works at each step/file

Show drafts ▾



Magic at your fingertips

Going beyond learning

◆ Go Library for Deploying Python Functions

This is a basic Golang library that allows users to deploy Python functions through Go. It utilizes subprocess execution and assumes the Python interpreter is available on the system.

Files:

1. deploy.go
2. utils.go

deploy.go

```
package deploy

import (
    "bytes"
    "encoding/json"
    "errors"
    "fmt"
    "io/ioutil"
    "os/exec"
)

// Function represents a Python function with its name
and code
type Function struct {
    Name string `json:"name"`
    Code string `json:"code"`
}
```

```
// DeployFunction takes a Python function and deploys it
func DeployFunction(f Function) (string, error) {
    // Create a temporary file for the Python code
    tmpFile, err := ioutil.TempFile("", "pyfunc-")
    if err != nil {
        return "", fmt.Errorf("failed to create temporary file:
%w", err)
    }
    defer os.Remove(tmpFile.Name())

    // Write the Python code to the temporary file
    if _, err := tmpFile.Write([]byte(f.Code)); err != nil {
        return "", fmt.Errorf("failed to write to temporary file:
%w", err)
    }

    // Build the command to execute the Python function
    cmd := exec.Command("python", tmpFile.Name())

    // Capture the output
    var out bytes.Buffer
    cmd.Stdout = &out

    // Execute the command and handle errors
    if err := cmd.Run(); err != nil {
        return "", fmt.Errorf("failed to execute Python function:
%w", err)
    }

    // Return the function output as a string
    return out.String(), nil
}
```

Experiment 3

Magic at your fingertips

Going beyond learning



Explanation:

- `Function` struct defines the data structure for storing the Python function name and code.
- `DeployFunction` takes a `Function` object and performs the deployment.
 - Creates a temporary file to hold the Python code.
 - Writes the code to the temporary file.
 - Builds a command to execute `python` with the temporary file path as an argument.
 - Captures the standard output of the command.
 - Executes the command and checks for errors.
 - Returns the captured output as a string, which represents the result of the Python function.

utils.go

```
package utils

import (
    "encoding/json"
    "fmt"
)

// ValidateFunction checks if the provided Python code is valid JSON
func ValidateFunction(f string) error {
    var funcData Function
    if err := json.Unmarshal([]byte(f), &funcData); err != nil {
        return fmt.Errorf("invalid function format: %w", err)
    }
    return nil
}
```

Magic at your fingertips

Going beyond learning

◆ Explanation:

- `ValidateFunction` takes a string representing the Python function and checks if it's valid JSON.
- This allows parsing the function code and name before deployment.

How to use?

```
import (
    "fmt"
    "./deploy"
)

func main() {
    // Example Python function
    pythonFunction := deploy.Function{
        Name: "add",
        Code: `def add(a, b):
            return a + b
        print(add(2, 3))`,
    }

    // Deploy the function
    output, err := deploy.DeployFunction(pythonFunction)
    if err != nil {
        fmt.Println("Error deploying function:", err)
        return
    }

    fmt.Println("Function output:", output)
}
```

Experiment 3

Magic at your fingertips

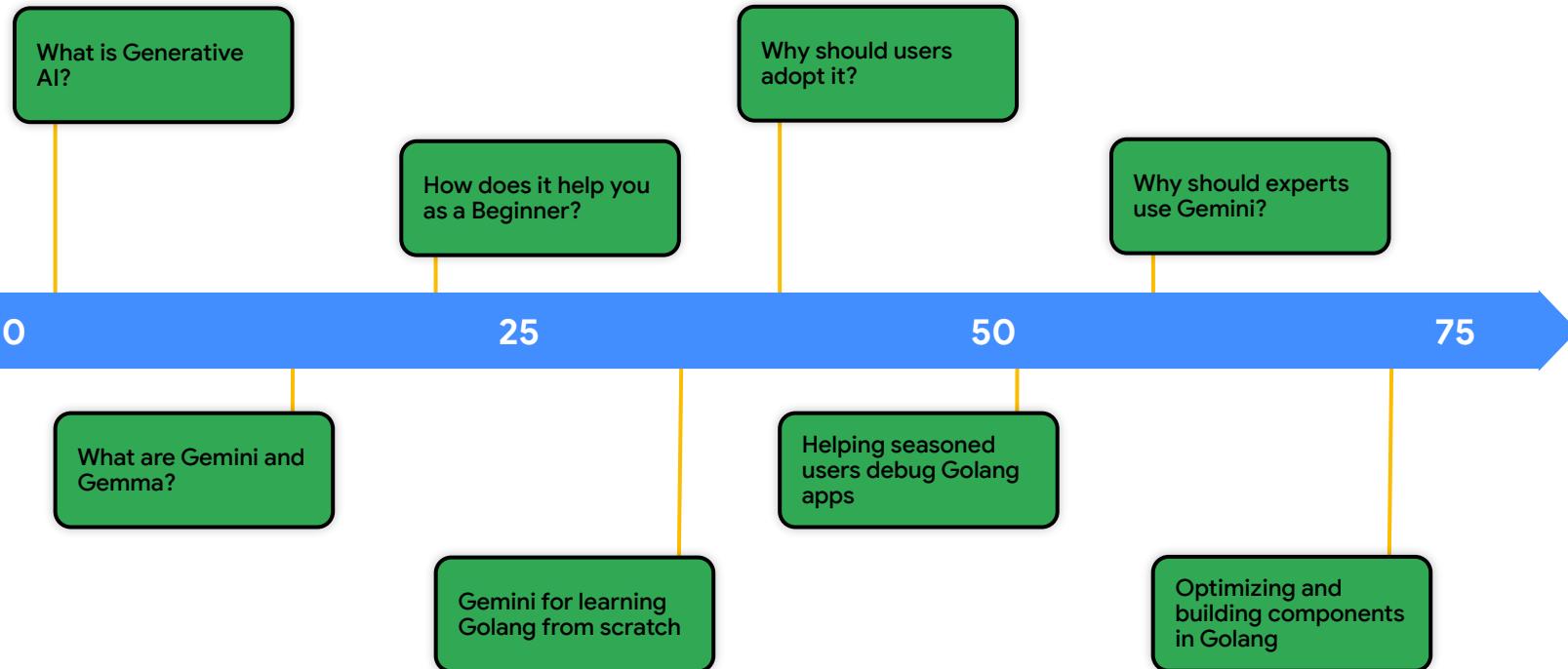
Going beyond learning

- ◆ This example demonstrates how to define a Python function, deploy it using the `DeployFunction` function, and retrieve the output.

Note:

- This is a basic example and can be extended to handle more complex scenarios like passing arguments, handling different data types, and error handling in the Python code itself.
- Security considerations are not included in this example. Ensure proper validation and sanitization of user-provided Python code before deployment.

Things to cover today

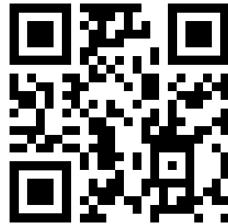


Thank you!

감사합니다!



Suvaditya Mukherjee
ML GDE | GDG MAD
@halcyonrayes



^ Slides here!

< Twitter

