../shinning-fiber/app.go

```go
package main

import (
 "html/template"
 "log"
 "os"
 "path/filepath"

 "github.com/gofiber/recipes/template-asset-bundling/handlers"

 "github.com/gofiber/fiber/v2"
 "github.com/gofiber/fiber/v2/middleware/logger"
 "github.com/gofiber/fiber/v2/middleware/recover"
 "github.com/gofiber/template/html/v2"
)

func main() {
 // Create view engine
 engine := html.New("./views", ".html")

 // Disable this in production
 engine.Reload(true)

 engine.AddFunc("getCssAsset", func(name string) (res template.HTML) {
  filepath.Walk("public/assets", func(path string, info os.FileInfo, err error) error {
   if err != nil {
    return err
   }
   if info.Name() == name {
    res = template.HTML("")
   }
   return nil
  })
  return
 })

 // Create fiber app
 app := fiber.New(fiber.Config{
  Views:       engine,
  ViewsLayout: "layouts/main",
 })

 // Middleware
 app.Use(recover.New())
 app.Use(logger.New())

 // Setup routes
 app.Get("/", handlers.Home)
 app.Get("/about", handlers.About)

 // Setup static files
 app.Static("/public", "./public")

 // Handle not founds
 app.Use(handlers.NotFound)

 // Listen on port 3000
 log.Fatal(app.Listen(":3000"))
}
```

../shinning-fiber/database/database.go

```go
package database

import (
 "log"
 "os"

 "gorm-mysql/models"

 "gorm.io/driver/mysql"
 "gorm.io/gorm"
)

var (
 DBConn *gorm.DB
)

// connectDb
func ConnectDb() {

 // refer https://github.com/go-sql-driver/mysql#dsn-data-source-name for details
 dsn := "user:pass@tcp(127.0.0.1:3306)/dbname?charset=utf8mb4&parseTime=True&loc=Local"
 /*
  NOTE:
  To handle time.Time correctly, you need to include parseTime as a parameter. (more parameters)
  To fully support UTF-8 encoding, you need to change charset=utf8 to charset=utf8mb4. See this article for a detailed explanation
 */

 db, err := gorm.Open(mysql.Open(dsn), &gorm.Config{})

 if err != nil {
  log.Fatal("Failed to connect to database. \n", err)
  os.Exit(2)
 }

 log.Println("connected")
 db.AutoMigrate(&models.Book{})
 DBConn = db

}
```

../shinning-fiber/handlers/handlers.go

```
package handlers

import (
 "github.com/gofiber/fiber/v2"
)

// Home renders the home view
func Home(c *fiber.Ctx) error {
 return c.Render("index", fiber.Map{
  "Title": "Hello, World!",
 })
}

// About renders the about view
func About(c *fiber.Ctx) error {
 return c.Render("about", nil)
}

// NoutFound renders the 404 view
func NotFound(c *fiber.Ctx) error {
 return c.Status(404).Render("404", nil)
}
```

../shinning-fiber/models/Book.go

```
package models

import "gorm.io/gorm"

// Book model
type Book struct {
 gorm.Model

 Title  string `json:"title"`
 Author string `json:"author"`
}
```

../shinning-fiber/routes/routes.go

```
package routes

import (
 "gorm-mysql/database"
 "gorm-mysql/models"
 "strconv"
 "github.com/gofiber/fiber/v2"
)

//Hello
func Hello(c *fiber.Ctx) error {
 return c.SendString("fiber")
}

//AddBook
func AddBook(c *fiber.Ctx) error {
 book := new(models.Book)
 if err := c.BodyParser(book); err != nil {
  return c.Status(400).JSON(err.Error())
 }

 database.DBConn.Create(&book)

 return c.Status(200).JSON(book)
}

func GetBook(c *fiber.Ctx) error {
 books := []models.Book{}

 database.DBConn.First(&books, c.Params("id"))

 return c.Status(200).JSON(books)
}

//AllBooks
func AllBooks(c *fiber.Ctx) error {
 books := []models.Book{}

 database.DBConn.Find(&books)

 return c.Status(200).JSON(books)
}

//Update
func Update(c *fiber.Ctx) error {
 book := new(models.Book)
 if err := c.BodyParser(book); err != nil {
  return c.Status(400).JSON(err.Error())
 }
 id, _ := strconv.Atoi(c.Params("id"))

 database.DBConn.Model(&models.Book{}).Where("id = ?", id).Update("title", book.Title)

 return c.Status(200).JSON("updated")
}

//Delete
func Delete(c *fiber.Ctx) error {
 book := new(models.Book)

 id, _ := strconv.Atoi(c.Params("id"))

 database.DBConn.Where("id = ?", id).Delete(&book)

 return c.Status(200).JSON("deleted")
}
```