I'll break down the application starting process file by file and explain how each component works together.

1. **Starting the Application - Main Entry Point**

First, in `cmd/main.go`:

go

```go
package main

import (
    "log"
    "net/http"
    "mcq-test-system/internal/handlers"
)

func main() {
    // Initialize handler
    testHandler := handlers.NewTestHandler()

    // Set up routes
    mux := http.NewServeMux()
    mux.HandleFunc("/", testHandler.StartTest)  // This handles the initial page
    mux.HandleFunc("/question", testHandler.HandleQuestion)
    mux.HandleFunc("/submit", testHandler.HandleSubmit)
    mux.HandleFunc("/report", testHandler.GenerateReport)

    // Serve static files (CSS)
    fs := http.FileServer(http.Dir("static"))
    mux.Handle("/static/", http.StripPrefix("/static/", fs))

    // Start server on port 8080
    log.Println("Server starting on :8080")
    log.Fatal(http.ListenAndServe(":8080", mux))
}
```

When you run `go run cmd/main.go`:

1. Creates new TestHandler
2. Sets up URL routes
3. Starts web server on port 8080
4. "/" route maps to testHandler.StartTest function
5. **Initial Page Load**

In `internal/handlers/test_handler.go`, the StartTest function handles the initial page:

go

```go
func NewTestHandler() *TestHandler {
    return &TestHandler{
        // Load all HTML templates when handler is created
        templates: template.Must(template.ParseGlob("templates/*.html")),
```

```
        sessions:  make(map[string]*models.TestSession),
    }
}

func (h *TestHandler) StartTest(w http.ResponseWriter, r *http.Request) {
    // If user just accessing the page (GET request)
    if r.Method == "GET" {
        // Show the start page
        h.templates.ExecuteTemplate(w, "start.html", nil)
        return
    }

    // If user submitted the form (POST request)
    numQuestions, _ := strconv.Atoi(r.FormValue("num_questions"))
    if numQuestions < 1 {
        http.Error(w, "Invalid number of questions", http.StatusBadRequest)
        return
    }

    // Load questions from Excel
    questions, err := utils.LoadQuestionsFromExcel("MCQ_question.xlsx")
    if err != nil {
        http.Error(w, "Failed to load questions", http.StatusInternalServerError)
        return
    }

    // Create new session
    sessionID := strconv.FormatInt(time.Now().UnixNano(), 10)
    session := &models.TestSession{
        Questions:    questions[:numQuestions],
        StartTime:    time.Now(),
        NumQuestions: numQuestions,
        Answers:      make([]int, numQuestions),
    }
    h.sessions[sessionID] = session

    // Set session cookie
    http.SetCookie(w, &http.Cookie{
        Name:  "session_id",
        Value: sessionID,
        Path:  "/",
    })

    // Redirect to first question
    http.Redirect(w, r, "/question", http.StatusSeeOther)
}
```

3. Start Page Template

The initial page is defined in `templates/start.html`:

```html
<!DOCTYPE html>
<html>
<head>
    <title>MCQ Test</title>
    <link rel="stylesheet" href="/static/css/style.css">
</head>
<body>
    <div class="container">
        <h1>MCQ Test</h1>
        <form method="POST">
            <div class="form-group">
                <label>Number of questions:</label>
                <input type="number" name="num_questions" required min="1" class="form-control">
                <small class="form-text">You will have 2 minutes per question</small>
            </div>
            <button type="submit" class="btn-primary">Start Test</button>
        </form>
    </div>
</body>
</html>
```

4. Question Loading Process

In `internal/utils/excel.go`:

```go
func LoadQuestionsFromExcel(filename string) ([]models.Question, error) {
    // Open Excel file
    f, err := excelize.OpenFile(filename)
    if err != nil {
        return nil, err
    }
    defer f.Close()

    // Get all rows
    rows, err := f.GetRows("Sheet1")
    if err != nil {
        return nil, err
    }

    var questions []models.Question
    // Skip header row (i=0)
    for i, row := range rows {
        if i == 0 {
            continue
```

```
        }

        correctAnswer, _ := strconv.Atoi(row[7])
        question := models.Question{
            RawText:       row[0],
            QuestionText:  row[1],
            Answers:       []string{row[2], row[3], row[4], row[5]},
            Difficulty:    row[6],
            CorrectAnswer: correctAnswer,
            Explanation:   row[8],
            Explanation2:  row[9],
            Subject:      row[10],
            Topic:        row[11],
        }
        questions = append(questions, question)
    }
    return questions, nil
}
```

Step by Step Flow:

1. User opens browser to `http://localhost:8080`
   - Web server receives request
   - main.go routes "/" to StartTest handler
   - StartTest sees GET request, shows start.html
2. User enters number of questions and submits
   - Browser sends POST request to "/"
   - StartTest handler:
     - Reads number from form
     - Loads questions from Excel
     - Creates new session
     - Sets session cookie
     - Redirects to first question
3. Question Loading:
   - Excel file opened
   - Rows converted to Question structs
   - Questions stored in session
   - Random questions selected based on user's requested number

Required Files for Excel:

Cc

```
MCQ_question.xlsx structure:
Column A: Raw_Text
Column B: Question_Text
Column C-F: Answer_1 through Answer_4
Column G: Difficulty
```

```
Column H: Correct_Answer (0-3)
Column I: Explanation
Column J: Explanation_2
Column K: Subject
Column L: Topic
```

Key Components Working Together:

1. `main.go`: Sets up server and routes
2. `test_handler.go`: Manages user sessions and request handling
3. `excel.go`: Loads and parses questions
4. `start.html`: User interface for starting test
5. `models/question.go`: Defines data structures
6. `MCQ_question.xlsx`: Stores actual questions

Would you like me to explain any specific part in more detail?