1. **Scenario Title: *College Fest Event Registration System***

Question :  
Draw the DFD (Level-0, and Level-1) for a College Fest Event Registration System using any suitable online tool.

**Aim:** To design and document a comprehensive Data Flow Diagram (DFD) for a Grocery Shopping System.

**Required Tools:** Computer, Google Docs, Draw.io (or any other diagramming tool)

### **Theory:**

A **Data Flow Diagram (DFD)** is a visual tool used to represent how data moves within a system. It is widely used during system analysis and design to depict **where data originates, how it is processed, and where it is stored or sent**.

A DFD consists of four key elements:

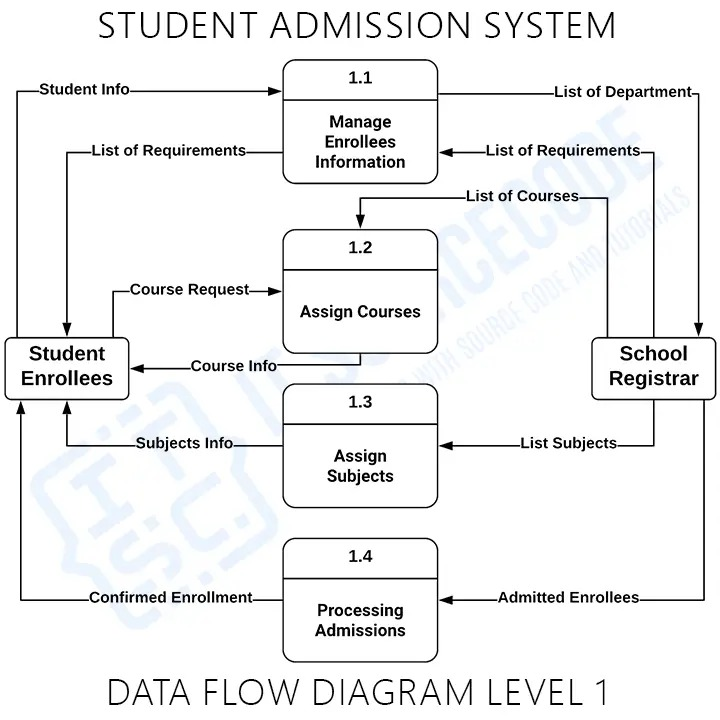
* **Processes:** Represent activities that transform input data into output data. Typically drawn as **rounded rectangles or circles**.
* **Data Stores:** Indicate **repositories** where data is kept within the system. Shown as **open-ended rectangles**.
* **Data Flows:** Arrows that **illustrate the direction of data movement** between entities, processes, and storage components.
* **External Entities:** Actors **outside the system** (like users or third-party services) that **send or receive data**. These are drawn as **plain rectangles**.

DFDs emphasize **data movement and interaction** rather than program logic or control flow, making them ideal for planning and documentation.

### **DFD Levels**

DFDs are developed in layers to show increasing detail.

#### **1. Level 0 (Context Diagram):**

* Gives a **single-process overview** of the entire system.
* Focuses on **system boundaries** and **interactions with external entities**.
* No internal processes are shown — only **data exchanges between the system and its users/services**.  
  **2. Level 1 (Decomposition Diagram):**
* Expands the main process into **multiple major functional blocks**.
* Displays **data stores** and **internal data exchanges**.
* Makes it easier to understand **system functions like browsing, ordering, payment, etc.**

q2. To design and document test cases for White Box Testing of a selected application module.

**REQUIREMENTS:**

* PC with Linux/Windows Operating System
* Google Chrome Browser
* Internet Connection
* Google Docs
* Google Colab

**THEORY:**

White Box Testing (also known as Structural Testing or Glass Box Testing) is a testing technique in which the internal logic, structure, and flow of the source code are examined to design effective test cases. It ensures that every path, condition, and loop within the code functions correctly.

### **1. Basic Path Testing**

Basic Path Testing is used to derive a set of **linearly independent execution paths** through the program’s **Control Flow Graph (CFG)**.  
 This technique uses **Cyclomatic Complexity**, denoted as:

**V(G) = D + 1**

Where : D = Number of decision points (like if, while, for, case statements)

**Given:**

def categorize(a: int, b: int) -> str:

"""

Toy function with a few decisions.

Expected cyclomatic complexity (V(G)) = 4

- if a > 0 -> 1

- if b % 2 == 0 (when a>0) -> 1

- if b > 10 (when a<=0) -> 1

Total decision points = 3; V(G) = decisions + 1 = 5

"""

if a > 0:

if b % 2 == 0:

return "A" # path 1

else:

return "B" # path 2

else:

if b > 10:

return "C" # path 3

return "D" # path 4

V(G) = D + 1 = 3 +1= **4**

That matches the same result you get from the Control Flow Graph method Cyclomatic Complexity : 3.

### **2. Structure Path Testing**

Structure Path Testing ensures that **all decision outcomes (True and False)** are covered during execution.  
 This method validates **every possible flow of control** in the program, ensuring correctness of internal logic and complete **condition coverage**.

**OBSERVATION:**

* The **Cyclomatic Complexity** of the given program was found to be **4**, indicating **three independent paths**.
* **Basic Path Testing** ensured that all executable statements and logical flows were covered at least once (Statement Coverage).
* **Structure Path Testing** validated both **True and False** outcomes for each decision (Condition Coverage), confirming correct program behavior.

**If this is the program, then your White Box Test Case Table would look like this:**

| **Test Case ID** | **Input: a** | **Input: b** | **Expected Output** | **Path Covered** | **Test Type** | **Description** | **Status** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **TC01** | **5** | **8** | **"A"** | **Path 1 (a>0 & b even)** | **Basic Path / Condition True–True** | **Tests when a is positive and b is even** | **✅ Pass** |
| **TC02** | **3** | **7** | **"B"** | **Path 2 (a>0 & b odd)** | **Basic Path / Condition True–False** | **Tests when a is positive and b is odd** | **✅ Pass** |
| **TC03** | **-1** | **15** | **"C"** | **Path 3 (a<=0 & b>10)** | **Basic Path / Nested Condition True** | **Tests when a is zero/negative and b is greater than 10** | **✅ Pass** |
| **TC04** | **-4** | **5** | **"D"** | **Path 4 (a<=0 & b<=10)** | **Basic Path / Nested Condition False** | **Tests when a is zero/negative and b is less or equal to 10** | **✅ Pass** |

**3.Scenario Title: *College Fest Event Registration System***

Question:  
Design and write test cases for Black Box Testing of the College Fest Event Registration System, covering major functionalities such as student registration, event selection, and confirmation generation using Equivalence Class Partitioning and Boundary Value Analysis techniques. Include test case tables specifying input, expected output, and test result criteria.

Ans:

**Theory:** Black Box Testing is a software testing method in which the functionality of the software is tested without looking into the internal code or logic. The tester provides input and observes the output.

**Techniques of Black Box Testing:**

1. **Equivalence Partitioning (EP):** Divides input data into valid and invalid partitions and selects representatives.
2. **Boundary Value Analysis (BVA):** Focuses on values at the boundaries since errors often occur there.

| **Technique** | **Description** |
| --- | --- |
| **ECP (Equivalence Class Partitioning)** | Input values divided into valid and invalid groups to reduce number of test cases |
| **BVA (Boundary Value Analysis)** | Focuses on edge/boundary values for higher defect detection |

## **Test Cases (ECP + BVA)**

### **A. Student Registration – Test Cases**

Parameters:

* Name: must not be empty
* Age: **Valid range = 18–35**
* Email: must be valid format

| **TC ID** | **Input (Name, Age, Email)** | **Technique** | **Expected Output** | **Result Criteria** |
| --- | --- | --- | --- | --- |
| SR01 | "Iqra", 20, "iqra@mail.com" | ECP Valid | Registration Successful | ✅ Pass |
| SR02 | "", 22, "abc@mail.com" | ECP Invalid | Error: Name Required | ✅ Pass |
| SR03 | "Aisha", 17, "a@mail.com" | BVA Below | Error: Age below limit (18) | ✅ Pass |
| SR04 | "Ali", 18, "a@mail.com" | BVA Boundary | Registration Successful | ✅ Pass |
| SR05 | "Zara", 36, "z@mail.com" | BVA Above | Error: Age exceeds limit (35) | ✅ Pass |
| SR06 | "Sam", 25, "wrongemail" | ECP Invalid | Error: Invalid Email Format | ✅ Pass |

### **B. Event Selection – Test Cases**

Rules:

* User must choose at least **1** event
* Max **5** events can be selected

| **TC ID** | **Input: Number of Events Selected** | **Technique** | **Expected Output** | **Result Criteria** |
| --- | --- | --- | --- | --- |
| ES01 | 1 | BVA (Lower boundary) | Event Selection Successful | ✅ Pass |
| ES02 | 5 | BVA (Upper boundary) | Event Selection Successful | ✅ Pass |
| ES03 | 0 | ECP Invalid | Error: Select at least one event | ✅ Pass |
| ES04 | 6 | ECP Invalid | Error: Max 5 events allowed | ✅ Pass |

### **C. Confirmation Generation – Test Cases**

Rules:

* Confirmation should be generated only if registration + event selection successful

| **TC ID** | **Registration Status** | **Event Selection Status** | **Output Expected** | **Technique** | **Result Criteria** |
| --- | --- | --- | --- | --- | --- |
| CG01 | Success | Success | Confirmation Generated | ECP Valid | ✅ Pass |
| CG02 | Failed | Success | No Confirmation: Fix Registration | ECP Invalid | ✅ Pass |
| CG03 | Success | Failed | No Confirmation: Fix Event Selection | ECP Invalid | ✅ Pass |

**4] Scenario Title: *College Fest Event Registration System***

Question:  
Using any standard project management or diagramming tool, design a Scrum Framework for the College Fest Event Registration System, showing key elements such as Product Backlog, Sprint Backlog, Scrum Roles (Product Owner, Scrum Master, Development Team) etc.

Steps:

# **Step-by-step: Create the Project & Team in Jira**

### **1. Sign in / sign up**

1. Go to your Jira Cloud site (yourcompany.atlassian.net) and sign in.
2. If you don’t have an account, create one and choose a Jira Cloud site (free tier available).  
    (If you’re using an organization/site, you may need a Jira admin to create projects for you.)

### **2. Create the project**

1. From the top navigation click **Projects → Create project**.
2. Choose a template. For Scrum choose **Scrum software development** (or **Team-managed → Scrum** for a simpler setup).
3. Pick **Company-managed** (more control) or **Team-managed** (simpler, per-team settings).
4. Enter **Project name** (e.g., *College Fest Event Registration*) and Project key, then **Create**.  
    (You need the right permissions to create projects — if the option is missing ask your Jira admin.) [Atlassian+1](https://www.atlassian.com/software/jira/guides/getting-started/basics?utm_source=chatgpt.com)

### **3. Create / verify the Scrum board**

* A Scrum template usually creates a board automatically. If not: **Projects → (Select project) → Board → Create board → Scrum board** and connect it to your project’s filter/epic.  
   You can also create a board with sample data for practice. [Atlassian Support](https://support.atlassian.com/jira/kb/create-a-jira-cloud-board-with-sample-data/?utm_source=chatgpt.com)

### **4. Configure project settings (quick essentials)**

1. Open **Project settings** (bottom-left in the project sidebar).
2. Review **Issue types** (Story, Task, Bug, Epic) and add any custom fields you need (e.g., Event Type, Seat Count).
3. Check **Workflows** if you want custom statuses (To Do → In Progress → Testing → Done).
4. Configure **Screens** and **Permissions** if required (who can create/transition issues, etc.).  
    (Company-managed projects give more options for shared workflows & permissions; team-managed keeps things simpler.) [Atlassian Support+1](https://support.atlassian.com/jira-cloud-administration/docs/create-and-edit-a-project/?utm_source=chatgpt.com)

# **Add team members & assign roles**

### **5. Invite users to the Jira site (if they’re not already users)**

1. From the Jira home, go to **Settings → User management → Invite users** (or **Teams → Invite people** in some UIs).
2. Enter the teammate’s email and send invite. They accept and get added to the site.  
    (Only site admins can add users to the Jira instance; project admins can add people to projects if users already exist.) [supademo.com+1](https://supademo.com/blog/guides/how-to-invite-a-teammate-in-jira/?utm_source=chatgpt.com)

### **6. Add people to the project & set their project role**

For **Team-managed** project:

1. Open the project → sidebar → **Project settings → Access** (or use the three-dot menu → **Add people**).
2. Type the user’s name or email.
3. From **Role** dropdown choose **Member**, **Viewer**, or **Administrator**. Click **Add**. [Atlassian Community+1](https://community.atlassian.com/learning/lesson/add-users-to-a-software-project?utm_source=chatgpt.com)

For **Company-managed** project:

1. Open project → **Project settings → People** (or **Users and roles**).
2. Click **Add people**, select user, then assign a **Project role** (e.g., Developer, Administrator).  
    Note: assigning some roles or changing groups may require Jira admin rights. [confluence.atlassian.com+1](https://confluence.atlassian.com/adminjiraserver/managing-project-roles-938847166.html?utm_source=chatgpt.com)

### **7. Recommended role assignments for Scrum**

* **Product Owner** → Project role: *Administrator* or *Product Owner* (manages backlog & priorities)
* **Scrum Master** → Project role: *Project admin / Member* (facilitates, removes blockers)
* **Development Team** → Project role: *Member / Developer* (create/transition issues)  
   Assign these roles by adding users to the project and selecting the corresponding role in the Add People dialog. [Atlassian](https://www.atlassian.com/software/jira/guides/permissions/overview?utm_source=chatgpt.com)

# **Set up backlog, sprints, and board workflow**

### **8. Populate Product Backlog**

1. In the project, go to **Backlog**.
2. Create **Epics** for big features (e.g., Student Registration, Event Management, Payment/Confirmation).
3. Add **Stories** or **Tasks** under Epics (e.g., “Student can register”, “Admin approve event”, “Send confirmation email”).

### **9. Create Sprint(s)**

1. From **Backlog**, click **Create sprint**.
2. Drag selected backlog items into the sprint.
3. During **Sprint Planning**, set sprint goal, start/end dates, and move the sprint to **Active** when the team starts work.

### **10. Daily use: update issues on the board**

* Team moves cards across columns (To Do → In Progress → Testing → Done).
* Use **Estimates** (story points) on stories and track sprint progress with the **Burndown chart** (Jira reports).

# **Extra useful steps & tips**

* **WIP & Swimlanes:** Use board settings → Swimlanes (by story, assignee, query) to organize work.
* **Permissions:** If you need fine-grained control over who can release, edit workflows, or manage versions, configure a **Permission Scheme** (company-managed only). [Atlassian](https://www.atlassian.com/software/jira/guides/permissions/overview?utm_source=chatgpt.com)
* **Notifications:** Configure project notifications so team members get emails for assignment, comments, or transitions.
* **Automation:** Use Jira automation rules (e.g., on issue moved to Done → send confirmation email).

# **Quick checklist (copyable)**

* Create Jira site / sign in
* Create Scrum project (company/team managed)
* Verify issue types and workflow
* Invite people to site (if needed)
* Add people to project → assign roles (Product Owner, Scrum Master, Devs)
* Create Epics & backlog items
* Create Sprint and start sprint planning
* Use board daily and track burndown

# **Short sample user story (so you can add one right away)**

**Epic:** Event Registration  
 **Story:** As a student, I want to register for an event so that I can attend the workshop.  
 **Acceptance Criteria:** Student fills form, age≥18, receives confirmation email, registration saved.

If you want, I can:

* Create a **copy-paste checklist** for your team to follow when you actually set up the project.
* Produce a **mock screenshot-style Jira board layout** (diagram) tailored to the College *Fest Event Registration System*.  
   Which would you like next?

Q5

1. **Scenario Title: *College Fest Event Registration System***

Question:  
Using any standard project management or diagramming tool, design a Kanban Board for the College Fest Event Registration System, showing workflow stages such as To Do, In Progress, Testing, and Done along with sample tasks under each stag

### **✅ 1️⃣ User Registration & Login**

This task focuses on enabling new users to sign up and existing users to securely access the system. It includes collecting necessary details like name, email, and mobile number, verifying credentials through OTP or email, and ensuring encrypted password storage for security. It also covers account recovery, session handling, and maintaining user profile data. The goal is to provide a smooth and secure login experience for all participants.

### **✅ 2️⃣ Event Management & Listing**

This task involves creating and managing all fest events within the system. Admins can add new events with details such as title, date, location, fees, and event category. Students can view a list of available events, filter by interest, and explore event-specific information. It ensures updated and visually clear event presentation so users can easily decide which events to participate in during the college festival.

### **✅ 3️⃣ Event Registration Process**

This task ensures students can register for events seamlessly. It includes checking real-time seat availability, generating unique registration IDs, and maintaining user-to-event mapping records. Users receive confirmation notifications after successful registration. It supports multiple event enrollments and prevents duplicate entries. The objective is to provide a simple, fast, and reliable registration procedure so participants can secure their spots for their preferred events.

### **✅ 4️⃣ Payment System Integration**

This task is responsible for enabling secure online payments for paid events. It integrates UPI, card, or wallet payment gateways to allow smoother transactions. The system verifies the payment status and generates digital receipts for the users. Admins can track revenue, and logs are stored for auditing. The aim is to ensure safe, transparent, and error-free payments, building user trust while simplifying the registration fee process.

### **✅ 5️⃣ Admin Dashboard & Control**

This task allows authorized college staff to manage both users and events efficiently. The admin can add, modify, or remove events, monitor registrations, and view overall statistics in real time. It also includes secure login for admins and options to export registration data for reports. The dashboard provides complete control and transparency over the fest workflow, improving decision-making and operational management.

### **✅ 6️⃣ Notification & Alert System**

The notification system keeps participants updated through emails or in-app alerts. It sends registration confirmations, schedule reminders, and important event changes. Alerts also help reduce confusion and missed activities during the fest. By automating timely messages, the system enhances communication between organizers and participants, ensuring that users always stay informed about their registered events and any updates from the organizers.

### **✅ 7️⃣ Feedback & Rating System**

This task gathers participant feedback after events to help improve future programs. Users can submit comments and rate each event based on their experience. The admin can analyze responses and review suggestions to identify strengths and areas needing improvement. Displaying event ratings also helps students make better participation decisions. It contributes to overall fest quality improvement by focusing on user satisfaction and engagement.

### **1️⃣ User Registration & Login**

**Sub-Task 1.1: Create Registration Form**

* ⬜ Add input fields (Name, Email, Phone)
* ⬜ Apply validation rules

**Sub-Task 1.2: Develop Secure Login**

* ⬜ Use encrypted password storage
* ⬜ Add “Forgot Password” support

### **2️⃣ Event Management & Listing**

**Sub-Task 2.1: Add New Event**

* ⬜ Enter event details in form
* ⬜ Upload event banner

**Sub-Task 2.2: Display Event List**

* ⬜ Create event cards UI
* ⬜ Enable search/filter options

### **3️⃣ Event Registration Process**

**Sub-Task 3.1: Seat Availability Check**

* ⬜ Update seat count after each registration

**Sub-Task 3.2: Registration Confirmation**

* ⬜ Auto-generate registration ID
* ⬜ Send confirmation message

### **4️⃣ Payment System Integration**

**Sub-Task 4.1: Online Payment Setup**

* ⬜ Connect UPI/Card payment API

**Sub-Task 4.2: Payment Receipt**

* ⬜ Save receipt in user profile

### **5️⃣ Admin Dashboard & Control**

**Sub-Task 5.1: Event Management**

* ⬜ Add/Edit/Delete event buttons

**Sub-Task 5.2: Registration Analytics**

* ⬜ Show total participants count chart

### **6️⃣ Notification & Alert System**

**Sub-Task 6.1: Confirmation Messages**

* ⬜ Send email on successful registration

**Sub-Task 6.2: Event Update Alerts**

* ⬜ Alert users for schedule changes

### **7️⃣ Feedback & Rating System**

**Sub-Task 7.1: Feedback Form**

* ⬜ Add comment and star rating fields

**Sub-Task 7.2: Admin Review**

* ⬜ Display average rating for each event

### **📌 Aim**

To design a Kanban Board for the *College Fest Event Registration System* by organizing project tasks into workflow stages such as **To Do**, **In Progress**, **Testing**, and **Done** using a standard project management tool.

### **📌 Theory**

A **Kanban Board** is a visual project management technique used to track the progress of work. Kanban focuses on continuous workflow improvement by visualizing tasks on a board. Each task is represented as a card and moves across different stages of the workflow from start to completion. This helps teams identify bottlenecks, manage workload, and ensure smooth task progress.

Kanban follows the concept of **"Just-In-Time" delivery**, reducing delays and improving team productivity.

### **📌 What is a Kanban Card?**

A **Kanban Card** represents a single task or work item.  
 Each card includes:

* Task name
* Assigned team member
* Priority
* Checklist (Sub-tasks)
* Due dates
* Progress tracking

Kanban cards move step-by-step through workflow columns to show status updates.

### **📌 Kanban Workflow Stages (Columns)**

| **Column Name** | **Purpose** |
| --- | --- |
| **To Do** | Tasks planned but not yet started |
| **In Progress** | Tasks currently being worked on |
| **Testing** | Tasks completed and under review / QA |
| **Done** | Tasks fully completed and accepted |

### **📌 Purpose of Kanban Board**

* To visualize the complete workflow
* To improve task tracking and accountability
* To reduce task delays and bottlenecks
* To enhance team collaboration and transparency
* To deliver project tasks faster and more efficiently

**6.Scenario Title: *College Fest Event Registration System***

Question:  
Using any standard project management or diagramming tool, design a Gantt Chart for the College Fest Event Registration System, showing major project phases such as Requirement Gathering, System Design, Development, Testing, and Deployment with appropriate tasks, durations, dependencies, and timelines.

Theory:

A **Gantt Chart** is a project management tool used to visually represent the schedule of a project. It displays project tasks along a timeline in the form of horizontal bars, showing when each task starts, how long it will take, and when it will finish. It also helps to identify task dependencies, track progress, and monitor overall project status.

✔ Major phases  
 ✔ Tasks & Sub-tasks  
 ✔ Durations (weeks)  
 ✔ Dependencies  
 ✔ Milestones  
 ✔ Timeline example  
 ✔ How to add Sub-tasks in tools like **GanttProject / GanttPRO / MS Project**

## **Gantt Chart for College Fest Event Registration System**

| **Project** | **College Fest Event Registration System** |
| --- | --- |
| Duration | 10 Weeks |
| Start Date | 01 Nov 2025 |
| End Date | 10 Jan 2026 |
| Methodology | Waterfall |



In text

| **Phase** | **Tasks / Sub-tasks** | **Duration** | **Start** | **End** | **Dependency** | **Milestone** |
| --- | --- | --- | --- | --- | --- | --- |
| **1️⃣ Requirement Gathering** | Collect Requirements | 1 Week | Week 1 | Week 1 | — | Requirement Document |
|  | Stakeholder Interviews | 1 Week | Week 1 | Week 2 | — | — |
|  | Finalize SRS Document | 1 Week | Week 2 | Week 3 | Interviews | ✅ **SRS Approval** |
| **2️⃣ System Design** |  | 1 Week | Week 3 | Week 4 | SRS Approval | — |
|  | Database Design (ER Diagram) | 1 Week | Week 4 | Week 5 | Architecture | — |
|  | UI/UX Wireframes | 1 Week | Week 4 | Week 5 | Parallel | ✅ **Design Freeze** |
| **3️⃣ Development** | Frontend Development | 2 Weeks | Week 5 | Week 7 | Design Freeze | — |
|  | Backend Development | 2 Weeks | Week 6 | Week 8 | DB Design | — |
|  | Integration of Modules | 1 Week | Week 7 | Week 8 | Frontend+Backend | ✅ **Development Complete** |
| **4️⃣ Testing** | Functional Testing | 1 Week | Week 8 | Week 9 | Dev Complete | — |
|  | Bug Fixing | 1 Week | Week 9 | Week 10 | Testing | ✅ **UAT Sign-off** |
| **5️⃣ Deployment** | Server Hosting | 2 Days | Week 10 | Week 10 | UAT Sign-off | — |
|  | Final Release | 1 Day | Week 10 | Week 10 | Hosting | ✅ **Project Go-Live 🎉** |

Project milestone

| **Milestone** | **Created In Week** |
| --- | --- |
| SRS Approval | Week 3 |
| Design Freeze | Week 5 |
| Development Complete | Week 8 |
| UAT Sign-off | Week 10 |
| Final Deployment | Week 10 |

7. **Scenario Title: *College Fest Event Registration System***

Question:  
Prepare a Risk Mitigation, Monitoring, and Management (RMMM) Plan for the College Fest Event Registration System, identify at least five potential project risks. Using appropriate tools or spreadsheets, calculate and tabulate key risk parameters such as Risk Probability, Impact, Risk Exposure, and Risk Priority etc. After preparing the table, explain your mitigation and contingency strategies for the top three critical risks on paper with proper justification.

Note : You can consider values by your own

### **✅ RMMM Plan**

**Scenario Title:** College Fest Event Registration System  
 **Objective:** Identify major risks during the development of the system and define strategies to mitigate, monitor, and manage them effectively.

## **Risk Identification Table with Risk Analysis**

*Note: Probability & Impact values are assumed on a scale of 1 to 5.* *Risk Exposure (RE) = Probability × Impact* *Risk Priority = Rank of RE score (High → Low)*

| Risk ID | Risk Description | Cause | Probability | Impact | Risk Exposure (RE) | Risk Priority |
| --- | --- | --- | --- | --- | --- | --- |
| R1 | Server/Hosting Failure during peak registration | High traffic overload, poor server capacity | 4 | 5 | 20 | 1 (Critical) |
| R2 | Data Security Breach (student information leak) | Weak authentication, unprotected database | 3 | 5 | 15 | 2 (Critical) |
| R3 | Development Delay | Limited resources, team availability issues | 4 | 3 | 12 | 3 (High) |
| R4 | Incorrect registration data from students | Missing validation, user input errors | 3 | 3 | 9 | 4 (Medium) |
| R5 | System usability issues (difficult UI) | Poor design, lack of usability testing | 2 | 3 | 6 | 5 (Low) |

## **Risk Mitigation & Contingency Strategies**

### **Risk R1: Server/Hosting Failure**

* **Mitigation Strategy:**
  + Use scalable cloud hosting (AWS/Azure)
  + Conduct load testing before launch
  + Maintain backup server handling high traffic
* **Contingency Plan:**
  + Emergency server switching
  + Allow offline registration temporarily
* **Justification:**
  + The system is used by many students at the same time; downtime causes frustration and loss of registrations.

### **Risk R2: Data Security Breach**

* **Mitigation Strategy:**
  + Enable HTTPS + Encryption for data storage
  + Strong authentication for admin access
  + Regular security audits and patch updates
* **Contingency Plan:**
  + Immediate blocking of vulnerable services
  + Notify affected users
* **Justification:**
  + Protecting student personal data is a primary ethical and legal requirement.

### **Risk R3: Development Delay**

* **Mitigation Strategy:**
  + Proper task scheduling using sprints
  + Regular progress monitoring (Scrum meetings)
  + Allocate backup resources if needed
* **Contingency Plan:**
  + Reduce non-critical features for initial release
  + Increase team collaboration to speed up tasks
* **Justification:**
  + Deadlines for college events cannot be postponed, so timely delivery is essential.

In excel file.

## **Conclusion**

The RMMM plan helps in:  
 ✔ Identifying risks early  
 ✔ Prioritizing based on severity  
 ✔ Defining preventive and backup actions  
 ✔ Ensuring smooth and secure event registration

This reduces uncertainty and improves the success rate of the project.