### ****What is a Synopsis of a Software Project?****

A **synopsis** is a brief summary of a software project that outlines its objectives, scope, features, technologies used, and expected outcomes. It acts as a blueprint for the project, providing a clear understanding of what the project aims to achieve and how it will be implemented.

### ****How to Write a Synopsis for a Software Project?****

A well-structured synopsis should include the following key sections:

#### **1. Title of the Project**

* Choose a meaningful title that reflects the project’s purpose.
* Example: **"Student Management System"**

#### **2. Introduction**

* Provide a brief introduction to the project.
* Mention why the project is necessary and what problem it solves.

#### **3. Objectives**

* Clearly define the primary goals of the project.
* Example:
  + To maintain student records efficiently.
  + To automate the process of student enrollment, attendance, and performance tracking.

#### **4. Scope of the Project**

* Define the boundaries of the project.
* Mention what features and functionalities will be included and any limitations.

#### **5. System Requirements**

* **Software Requirements:** Operating System, Database, Programming Language, IDE, etc.
* **Hardware Requirements:** Processor, RAM, Storage, etc.

#### **6. Modules of the Project**

* Divide the project into various modules. Example:
  + **Admin Module** – Manage students, teachers, and reports.
  + **Student Module** – View attendance, marks, and personal details.
  + **Teacher Module** – Update student grades, attendance, and records.

#### **7. Technologies Used**

* List the technologies used in development, such as:
  + Frontend: HTML, CSS, JavaScript
  + Backend: PHP, Java, or Python
  + Database: MySQL

#### **8. Functional Requirements**

* Specify the core functionalities, such as:
  + Add, Edit, Delete Student Records
  + Manage Attendance
  + Generate Reports

#### **9. Conclusion**

* Summarize how the project will be beneficial and its impact.

### ****Example: Synopsis for "Student Management System"****

#### **Title:**

**Student Management System**

#### **Introduction:**

The Student Management System is a web-based application designed to handle student-related data, including personal details, academic performance, attendance records, and more. It aims to digitize and automate the management of student information for schools, colleges, and universities.

#### **Objectives:**

* To simplify student record management.
* To automate attendance tracking and performance evaluation.
* To reduce manual work and improve efficiency.

#### **Scope of the Project:**

The system will allow:

* Administrators to add, update, and delete student details.
* Teachers to manage attendance and academic records.
* Students to view their attendance, grades, and academic progress.

#### **System Requirements:**

**Software Requirements:**

* OS: Windows/Linux
* Language: PHP, Java, or Python
* Database: MySQL

**Hardware Requirements:**

* Processor: Intel i3 or above
* RAM: 4GB or more

#### **Modules of the Project:**

1. **Admin Module**
   * Add/Delete/Update students and teachers.
   * Manage database records.
2. **Student Module**
   * View attendance and grades.
   * Access personal details.
3. **Teacher Module**
   * Mark attendance.
   * Enter grades.

#### **Technologies Used:**

* Frontend: HTML, CSS, JavaScript
* Backend: PHP
* Database: MySQL

#### **Functional Requirements:**

* Student registration and profile management.
* Attendance tracking system.
* Grade management and report generation.

#### **Conclusion:**

This system will improve efficiency by automating student data management, reducing paperwork, and enhancing accessibility for students, teachers, and administrators.

### ****System Design and SDLC for Student Management System****

When developing a **Student Management System (SMS)**, we follow **System Design** and **Software Development Life Cycle (SDLC)** methodologies to ensure a structured approach to building the software.

## **1. System Design for Student Management System**

### ****A. Architectural Design (High-Level Design)****

The **Student Management System** is typically designed using a **three-tier architecture**:

1. **Presentation Layer (Frontend):**
   * Handles the user interface (UI)
   * Technologies: HTML, CSS, JavaScript, Bootstrap
2. **Business Logic Layer (Backend):**
   * Processes requests and implements the logic
   * Technologies: PHP, Java, or Python
3. **Data Layer (Database):**
   * Stores student records, attendance, and results
   * Database: MySQL

### ****B. Database Design (ER Diagram & Tables)****

#### **ER Diagram:**

The **Entity-Relationship (ER) Diagram** helps in structuring the database.  
Entities:

* **Student** (id, name, email, class, phone, address)
* **Teacher** (id, name, subject, email, phone)
* **Class** (id, name, section)
* **Attendance** (id, student\_id, date, status)
* **Marks** (id, student\_id, subject, marks)

#### **Tables in the Database:**

| **Table Name** | **Fields** |
| --- | --- |
| students\_tbl | id, name, email, class, phone, address |
| teachers\_tbl | id, name, subject, email, phone |
| classes\_tbl | id, name, section |
| attendance\_tbl | id, student\_id, date, status |
| marks\_tbl | id, student\_id, subject, marks |

### ****C. Functional Design (Use Case Diagram)****

The **Use Case Diagram** defines different users and their interactions.

#### **Actors:**

1. **Admin:** Manages student and teacher records.
2. **Teacher:** Marks attendance, assigns grades.
3. **Student:** Views grades and attendance.

#### **Use Cases:**

* Admin: Add/Delete Students, Manage Teachers
* Teacher: Mark Attendance, Assign Marks
* Student: View Attendance, View Marks

## **2. Software Development Life Cycle (SDLC) for SMS**

The **Software Development Life Cycle (SDLC)** is a process used to design and develop software efficiently. The common SDLC models include **Waterfall, Agile, Spiral**, etc. For a **Student Management System**, the **Waterfall Model** or **Agile Model** is commonly used.

### ****Phases of SDLC for SMS:****

### ****1. Requirement Analysis****

* Understanding the needs of schools/colleges.
* Identifying modules: Student Registration, Attendance, Marks Management.

### ****2. Planning****

* Deciding project scope, cost, timeline.
* Choosing technology: PHP/MySQL, Java, Python.

### ****3. System Design****

* Creating **ER Diagrams, Use Case Diagrams**.
* Designing **Database Schema**.

### ****4. Implementation (Coding & Development)****

* Writing frontend (HTML, CSS, JavaScript).
* Developing backend (PHP/Python with MySQL).
* Implementing **CRUD (Create, Read, Update, Delete) Operations**.

### ****5. Testing****

* **Unit Testing:** Checking individual modules.
* **Integration Testing:** Testing the interaction between modules.
* **User Testing:** Checking if students and teachers can use the system properly.

### ****6. Deployment****

* Installing on school servers.
* Hosting on cloud platforms.

### ****7. Maintenance & Updates****

* Fixing bugs.
* Adding new features based on user feedback.

### ****Conclusion****

By following **System Design** and **SDLC**, we ensure that the **Student Management System** is built efficiently, is user-friendly, and meets the needs of schools or colleges.

### ****1. ER Diagram (Entity-Relationship Diagram)****

## **E-R Diagram – Student Management system**

E-R (Entity-Relationship) Diagram is used to represents the relationship between entities in a table. ER diagrams represent the logical structure of databases. ER Diagram represent relationship between two database tables.

E-R diagram means Entity Relationship diagram. Entity is a object of system, generally we refer entity as database table , the e-r diagram represent the relationship between each table of database. E-R diagram represent entity with attributes, attributes is a properties of entity. If we assume entity is a database table then all the columns of table are treat as attributes.

## **ER Diagram**

**Entity :** Entities are represented by **rectangle**. All table of database are treat as entity.

**Attributes :** Attributes are represented by **ellipses**. Attributes are properties of entities.

### ER Diagram Symbols

The **ER Diagram** represents how different entities (tables) relate to each other in the system.

#### **Entities & Relationships:**

* **Student** (id, name, email, class\_id, phone, address) → Belongs to **Class**
* **Teacher** (id, name, subject, email, phone) → Teaches **Class**
* **Class** (id, name, section) → Has **Students**
* **Attendance** (id, student\_id, date, status) → Belongs to **Student**
* **Marks** (id, student\_id, subject, marks) → Belongs to **Student**

### ****2. Use Case Diagram****

A **Use Case Diagram** shows how different users (Admin, Teacher, Student) interact with the system.

#### **Actors:**

1. **Admin:** Manages students, teachers, and classes.
2. **Teacher:** Marks attendance and enters student grades.
3. **Student:** Views attendance and grades.

#### **Use Cases:**

* Admin: Add/Delete Students, Manage Teachers
* Teacher: Mark Attendance, Assign Marks
* Student: View Attendance, View Marks

### ****3. Flowchart (Student Registration Process)****

A **flowchart** helps in understanding how student registration works in the system.

#### **Flow:**

1. Student fills the registration form.
2. Admin verifies details.
3. If details are valid → Student is added to the system.
4. If details are incorrect → Show error message.

### ****1. Entities & Attributes****

#### **1.1 Student Table**

* student\_id (Primary Key)
* name
* email
* phone
* address
* date\_of\_birth
* class\_id (Foreign Key → Class)

#### **1.2 Class Table**

* class\_id (Primary Key)
* class\_name
* section
* teacher\_id (Foreign Key → Teacher)

#### **1.3 Teacher Table**

* teacher\_id (Primary Key)
* name
* email
* phone
* subject

#### **1.4 Attendance Table**

* attendance\_id (Primary Key)
* student\_id (Foreign Key → Student)
* class\_date
* status (Present/Absent)

#### **1.5 Marks Table**

* marks\_id (Primary Key)
* student\_id (Foreign Key → Student)
* subject
* marks\_obtained
* exam\_date

#### **1.6 Fees Table**

* fee\_id (Primary Key)
* student\_id (Foreign Key → Student)
* amount
* due\_date
* payment\_status

### ****2. Relationships****

1. **A Student belongs to one Class**, but a **Class has many Students** (One-to-Many).
2. **A Class is assigned one Teacher**, but a **Teacher can handle multiple Classes** (One-to-Many).
3. **A Student has multiple Attendance records**, but each **Attendance record belongs to one Student** (One-to-Many).
4. **A Student has multiple Marks entries**, but each **Marks entry is linked to one Student** (One-to-Many).
5. **A Student has multiple Fee Payments**, but each **Fee record is for one Student** (One-to-Many).

### ****How to Create an ER Diagram in MySQL Workbench (Step-by-Step Guide)****

Creating an **Entity-Relationship (ER) Diagram** in **MySQL Workbench** is essential for designing a database visually. Follow this detailed step-by-step guide to create an **ER Diagram** for a **Student Management System**.

## **Step 1: Open MySQL Workbench**

* **Launch** MySQL Workbench from your system.
* Ensure that you have **MySQL Server** installed and running.

## **Step 2: Create a New EER Model**

1. Click on **File** → **New Model** (or press Ctrl + N).
2. A new window will open with an empty **EER (Enhanced Entity-Relationship) Model**.

## **Step 3: Create a New Database Schema (Optional)**

1. Click on **Database** → **Connect to Database** (Ctrl + U).
2. Enter your **MySQL credentials** and select an existing database (or create a new one).
3. Click **OK**.

## **Step 4: Open the EER Diagram**

1. In the **Model Overview** panel, click on **"Add Diagram"**.
2. A blank workspace for the **EER Diagram** will open.

## **Step 5: Add Tables (Entities)**

Now, we will **create tables** that represent entities in the Student Management System.

### ****Steps to Add a Table:****

1. Click on the **"Table" (Rectangle) Icon** in the toolbar.
2. Click anywhere on the **EER Diagram workspace** to place a new table.
3. Double-click on the table to edit its details:
   * Change the **Table Name** (e.g., students).
   * Add **columns** (e.g., student\_id, name, email, etc.).
   * **Set Primary Key (PK)** and **Foreign Keys (FK)**.
   * Choose **Data Types** (e.g., INT, VARCHAR(255), DATE).

### ****Example: Tables and Attributes****

#### **1. Students Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| student\_id | INT | PRIMARY KEY, AUTO\_INCREMENT |
| name | VARCHAR(255) | NOT NULL |
| email | VARCHAR(255) | UNIQUE |
| phone | VARCHAR(15) | NOT NULL |
| address | TEXT | NULL |
| date\_of\_birth | DATE | NULL |
| class\_id | INT | FOREIGN KEY (References classes.class\_id) |

#### **2. Classes Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| class\_id | INT | PRIMARY KEY, AUTO\_INCREMENT |
| class\_name | VARCHAR(100) | NOT NULL |
| section | VARCHAR(10) | NULL |
| teacher\_id | INT | FOREIGN KEY (References teachers.teacher\_id) |

#### **3. Teachers Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| teacher\_id | INT | PRIMARY KEY, AUTO\_INCREMENT |
| name | VARCHAR(255) | NOT NULL |
| email | VARCHAR(255) | UNIQUE |
| phone | VARCHAR(15) | NOT NULL |
| subject | VARCHAR(100) | NULL |

#### **4. Attendance Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| attendance\_id | INT | PRIMARY KEY, AUTO\_INCREMENT |
| student\_id | INT | FOREIGN KEY (References students.student\_id) |
| class\_date | DATE | NOT NULL |
| status | ENUM('Present', 'Absent') | NOT NULL |

#### **5. Marks Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| marks\_id | INT | PRIMARY KEY, AUTO\_INCREMENT |
| student\_id | INT | FOREIGN KEY (References students.student\_id) |
| subject | VARCHAR(100) | NOT NULL |
| marks\_obtained | INT | NOT NULL |
| exam\_date | DATE | NULL |

#### **6. Fees Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| fee\_id | INT | PRIMARY KEY, AUTO\_INCREMENT |
| student\_id | INT | FOREIGN KEY (References students.student\_id) |
| amount | DECIMAL(10,2) | NOT NULL |
| due\_date | DATE | NULL |
| payment\_status | ENUM('Paid', 'Pending') | NOT NULL |

## **Step 6: Define Relationships**

Now that tables are created, we need to **define relationships** between them.

### ****Steps to Create Relationships****

1. Click on the **"One-to-Many" (Crow’s Foot) Relationship Tool** in the toolbar.
2. Click on the **Primary Table** (e.g., classes).
3. Drag to the **Foreign Table** (e.g., students).
4. The **relationship line** appears automatically.

### ****Relationships in the ER Diagram****

1. **One Class has Many Students** → (students.class\_id references classes.class\_id)
2. **One Teacher teaches Many Classes** → (classes.teacher\_id references teachers.teacher\_id)
3. **One Student has Many Attendance Records** → (attendance.student\_id references students.student\_id)
4. **One Student has Many Marks Records** → (marks.student\_id references students.student\_id)
5. **One Student has Many Fee Payments** → (fees.student\_id references students.student\_id)

## **Step 7: Save and Export**

### ****Save the ER Diagram****

1. Click on **File** → **Save Model As**.
2. Choose a location and **save** your model.

### ****Export ER Diagram as an Image****

1. Click on **File** → **Export** → **Export as PNG**.
2. Choose a location and **save** the image.
3. You can now use this **ER Diagram** in documentation or presentations.

## **Final ER Diagram Structure**

Here is how your **ER Diagram** should look after following these steps:

scss

CopyEdit

STUDENTS (student\_id) ----< CLASSES (class\_id) ---- TEACHERS (teacher\_id)

| |

| |

| MARKS (marks\_id)

|

ATTENDANCE (attendance\_id)

|

FEES (fee\_id)

## **Conclusion**

## **Data Flow Diagram (DFD) for Student Management System**

A **Data Flow Diagram (DFD)** represents the **flow of data** in a system, showing how **input data** is transformed into **output data** through processes. It helps in understanding the system's functionality visually.

## **Levels of DFD**

DFDs are created at different levels:

1. **Level 0 (Context Diagram)** – Represents the entire system as a **single process**.
2. **Level 1 (Top-Level DFD)** – Shows the **major processes** in the system.
3. **Level 2 (Detailed DFD)** – Breaks down Level 1 processes into **sub-processes**.

## **Level 0 DFD (Context Diagram)**

### ****Explanation****

* Level 0 DFD is a **high-level representation** of the system.
* The **Student Management System (SMS)** is represented as a **single process**.
* It interacts with **external entities** (users) such as:
  + **Admin** (Manages Students, Teachers, Classes)
  + **Teachers** (Manage Attendance, Marks)
  + **Students** (View Results, Attendance)
* Data flows **between external entities and the system**.

### ****Diagram Representation****

+------------------------------------------------------+

| Student Management System |

| |

| +-----------------+ +------------------+ |

| | Admin |<---> | SMS Database | |

| +-----------------+ +------------------+ |

| +-----------------+ +------------------+ |

| | Teacher |<---> | SMS Database | |

| +-----------------+ +------------------+ |

| +-----------------+ +------------------+ |

| | Student |<---> | SMS Database | |

| +-----------------+ +------------------+ |

+------------------------------------------------------+

### ****Entities and Data Flow****

* **Admin**: Manages students, teachers, and classes.
* **Teacher**: Updates attendance, marks.
* **Student**: Views attendance, marks.
* **SMS Database**: Stores and retrieves information.

## **Level 1 DFD (Top-Level Diagram)**

### ****Explanation****

At this level, we break down the **main process** (Student Management System) into **sub-processes**:

1. **Manage Students** – Add, update, delete students.
2. **Manage Teachers** – Add, update, delete teachers.
3. **Manage Classes** – Assign teachers to classes.
4. **Manage Attendance** – Record student attendance.
5. **Manage Marks** – Store and retrieve marks.
6. **Generate Reports** – View attendance, marks reports.

### ****Diagram Representation****

+------------------------------------------------------+

| Student Management System |

| +----------------------------------------------+ |

| | 1. Manage Students (Admin) | |

| | - Add Student | |

| | - Update/Delete Student | |

| +----------------------------------------------+ |

| +----------------------------------------------+ |

| | 2. Manage Teachers (Admin) | |

| | - Add Teacher | |

| | - Update/Delete Teacher | |

| +----------------------------------------------+ |

| +----------------------------------------------+ |

| | 3. Manage Classes (Admin) | |

| | - Assign Teachers | |

| | - View Class Details | |

| +----------------------------------------------+ |

| +----------------------------------------------+ |

| | 4. Manage Attendance (Teacher) | |

| | - Record Attendance | |

| | - Update Attendance | |

| +----------------------------------------------+ |

| +----------------------------------------------+ |

| | 5. Manage Marks (Teacher) | |

| | - Enter Marks | |

| | - View Marks | |

| +----------------------------------------------+ |

| +----------------------------------------------+ |

| | 6. Generate Reports (Student, Admin) | |

| | - Attendance Report | |

| | - Marks Report | |

| +----------------------------------------------+ |

+------------------------------------------------------+

### ****Data Flow Between Processes****

1. **Admin Inputs**
   * Adds/Deletes Students, Teachers, Classes.
   * Updates database.
2. **Teacher Inputs**
   * Adds Attendance, Marks.
   * Updates the database.
3. **Student Inputs**
   * Views Reports (Attendance, Marks).

## **Level 2 DFD (Detailed Diagram)**

### ****Explanation****

At this level, each **process is further broken down** into sub-processes.

#### **Example: Manage Attendance Process**

* **Teacher Logs In**
* **Selects Class**
* **Records Attendance**
* **Saves Attendance**
* **Database Updates Attendance Table**
* **Student Views Attendance**

### ****Diagram Representation****

+------------------------------------------------------+

| Manage Attendance |

| +----------------------+ |

| | 1. Teacher Logs In | |

| +----------------------+ |

| +----------------------+ |

| | 2. Select Class | |

| +----------------------+ |

| +----------------------+ |

| | 3. Record Attendance | ---> SMS Database |

| +----------------------+ |

| +----------------------+ |

| | 4. Save Attendance | ---> Update Database |

| +----------------------+ |

| +----------------------+ |

| | 5. Student Views Attendance | |

| +----------------------+ |

+------------------------------------------------------+

### ****Data Flow****

1. **Teacher logs in** → System verifies credentials.
2. **Teacher selects class** → Fetches student list.
3. **Teacher records attendance** → System updates the database.
4. **Student views attendance** → Retrieves data from the database.

## **Conclusion**

### ****Key Takeaways****

✅ **Level 0 (Context Diagram)** → Overview of the system.  
✅ **Level 1 (Top-Level DFD)** → Major processes.  
✅ **Level 2 (Detailed DFD)** → Breakdown of each process.