Nama : Kezia S. F. Gultom

NIM : 12030123130205

Kelas : C-Akuntansi

UTS ANALISIS DESAIN SISTEM

Study Case

CaSe In POInt 4.2:

Deep river College Deep River College is a two-year school in Southern California. Twice a year, the fund-raising office at Deep River mails requests for donations to the alumni. The staff uses a word process ing program and a personal information database to create personalized letters. Data on past contributions and other alumni information, however, is stored manually. The dean, Alexandra Ali, recently submitted a systems request asking the college’s IT department to develop a computerized alumni information system. The school does not have a formal systems review committee, and each department has an individual budget for information services. Eddie Bateman, a systems analyst, performed a preliminary investigation and he concluded that the system met all the feasibility tests. After reading his report, Alexandra asked him to proceed with the systems analysis phase. Eddie has scheduled an interview with her, and he has asked you to help him prepare for the meeting. Specifically, he wants you to list all the topics he should cover during the interview. Eddie also wants you to pre pare a list of specific questions that he should ask. Be sure to include open-ended, closed ended, and range-of-response questions.

1. DFD

%%{ init: { "theme": "default" } }%%

graph TD;

A[Alumni Database] -->|Updates| B[Fundraising Staff]

A -->|Queries| C[Report Generation]

B -->|Donation Requests| D[Alumni]

C -->|Reports| B

D -->|Donations| A

subgraph Level\_1 [Level 1: System Overview]

direction TB

E[Manage Alumni Data] -->|Input Data| F[Alumni Database]

F -->|Store Data| G[Alumni Records]

G -->|Retrieve Data| E

end

subgraph Level\_2 [Level 2: Data Management]

direction TB

H[Input Alumni Info] -->|Collect Info| I[Data Entry Form]

I -->|Validate Info| J[Data Validation Process]

J -->|Store Validated Data| K[Alumni Database]

K -->|Generate Alumni Profiles| L[Profile Management]

end

subgraph Level\_3 [Level 3: Reporting]

direction TB

M[Generate Reports] -->|Select Criteria| N[Report Parameters]

N -->|Fetch Data| O[Data Retrieval Process]

O -->|Generate Report| P[Report Output]

P -->|Display Results| Q[Report Viewer]

end

subgraph Level\_4 [Level 4: User Management]

direction TB

R[User Authentication] -->|Login Process| S[Access Control]

S -->|Check Roles| T[Role Management]

T -->|Assign Permissions| U[Database Access]

U -->|Grant Access| V[User Dashboard]

end

subgraph Level\_5 [Level 5: Data Entry Process]

direction TB

W[Data Input Interface] -->|Collect Data| X[User Input]

X -->|Perform Data Checks| Y[Data Validation]

Y -->|Handle Errors| Z[Error Notification]

Z -->|Return to Input| W

Y -->|Pass Valid Data| AA[Data Storage]

end

1. ERD

erDiagram

ALUMNI {

int Alumni\_ID PK

string First\_Name

string Last\_Name

string Email

string Phone\_Number

int Graduation\_Year

string Degree

string Major

string Address

}

DONATION {

int Donation\_ID PK

decimal Amount

date Donation\_Date

string Purpose

int Alumni\_ID FK

}

USER {

int User\_ID PK

string Username

string Password

string Role

string Email

date Last\_Login

}

REPORT {

int Report\_ID PK

string Report\_Type

date Generation\_Date

int Generated\_By FK

}

EVENT {

int Event\_ID PK

string Event\_Name

date Event\_Date

string Location

string Description

}

EVENT\_ATTENDANCE {

int Attendance\_ID PK

int Alumni\_ID FK

int Event\_ID FK

}

ALUMNI ||--o{ DONATION : makes

USER ||--o{ REPORT : generates

ALUMNI ||--o{ EVENT\_ATTENDANCE : attends

EVENT ||--o{ EVENT\_ATTENDANCE : has

1. UML Diagram

**PlantUML Class Diagram Code**

plantuml

@startuml

class Alumni {

+int Alumni\_ID

+string First\_Name

+string Last\_Name

+string Email

+string Phone\_Number

+int Graduation\_Year

+string Degree

+string Major

+string Address

+makeDonation(amount: decimal, purpose: string)

+attendEvent(eventID: int)

+updateProfile()

+getDonationHistory()

}

class Donation {

+int Donation\_ID

+decimal Amount

+date Donation\_Date

+string Purpose

+getDonationDetails()

}

class User {

+int User\_ID

+string Username

+string Password

+string Role

+string Email

+login()

+generateReport(reportType: string)

+resetPassword()

}

class Report {

+int Report\_ID

+string Report\_Type

+date Generation\_Date

+generateSummary()

}

class Event {

+int Event\_ID

+string Event\_Name

+date Event\_Date

+string Location

+string Description

+registerAlumni(alumniID: int)

+getEventDetails()

}

class EventAttendance {

+int Attendance\_ID

+date Attendance\_Date

}

Alumni "1" --> "0..\*" Donation : makes

User "1" --> "0..\*" Report : generates

Alumni "1" --> "0..\*" EventAttendance : attends

Event "1" --> "0..\*" EventAttendance : has

@enduml

**Expanded UML Sequence Diagrams**

**Sequence Diagram 1: Making a Donation**

This diagram represents the process when an alumni makes a donation.

mermaid

Copy code

sequenceDiagram

participant Alumni

participant DonationService

participant AlumniDatabase

participant NotificationService

Alumni->>DonationService: makeDonation(amount, purpose)

DonationService->>AlumniDatabase: saveDonation(alumniID, amount, purpose)

AlumniDatabase-->>DonationService: Donation saved

DonationService->>NotificationService: sendAcknowledgment(alumniID)

NotificationService-->>Alumni: Donation acknowledgment sent

**Sequence Diagram 2: Registering for an Event**

This diagram illustrates how an alumni registers for an event.

mermaid

Copy code

sequenceDiagram

participant Alumni

participant EventService

participant EventDatabase

participant NotificationService

Alumni->>EventService: registerForEvent(eventID)

EventService->>EventDatabase: saveAttendance(alumniID, eventID)

EventDatabase-->>EventService: Attendance saved

EventService->>NotificationService: sendEventConfirmation(alumniID, eventID)

NotificationService-->>Alumni: Event confirmation sent

**Sequence Diagram 3: Generating a Report**

This diagram represents the process of generating a report by a user.

mermaid

Copy code

sequenceDiagram

participant User

participant ReportService

participant ReportDatabase

User->>ReportService: generateReport(reportType)

ReportService->>ReportDatabase: fetchReportData(reportType)

ReportDatabase-->>ReportService: return report data

ReportService-->>User: displayReport(reportData)

**PlantUML Use Case Diagram Code**

plantuml

Copy code

@startuml

left to right direction

actor Alumni

actor FundraisingStaff as Staff

actor Admin

usecase "Make Donation" as UC1

usecase "Attend Event" as UC2

usecase "View Reports" as UC3

usecase "Update Profile" as UC4

usecase "Send Donation Request" as UC5

usecase "Generate Reports" as UC6

usecase "View Donation History" as UC7

usecase "Manage Alumni Records" as UC8

usecase "Manage Events" as UC9

usecase "View All Reports" as UC10

usecase "Manage User Accounts" as UC11

Alumni --> UC1

Alumni --> UC2

Alumni --> UC3

Alumni --> UC4

Staff --> UC5

Staff --> UC6

Staff --> UC7

Admin --> UC8

Admin --> UC9

Admin --> UC10

Admin --> UC11

UC1 --> (Send Acknowledgment)

UC2 --> (Send Event Confirmation)

@enduml