

# Resume Generator

Sourabh Ghurye (130050001)

Nikhil Vyas (130050023)

Utkarsh Mall(130050037)

## Introduction

The application generates LaTeX based *Resume* for students. Student can create new resume or can view/edit resumes already saved to his/her account. The users can select themes for resume or can select the resume type. Students will also upload documents required for the verification. Placement Coordinators can view submitted resumes and documents on their interface. Once verification is done, users will be notified of it.

## ER Diagram

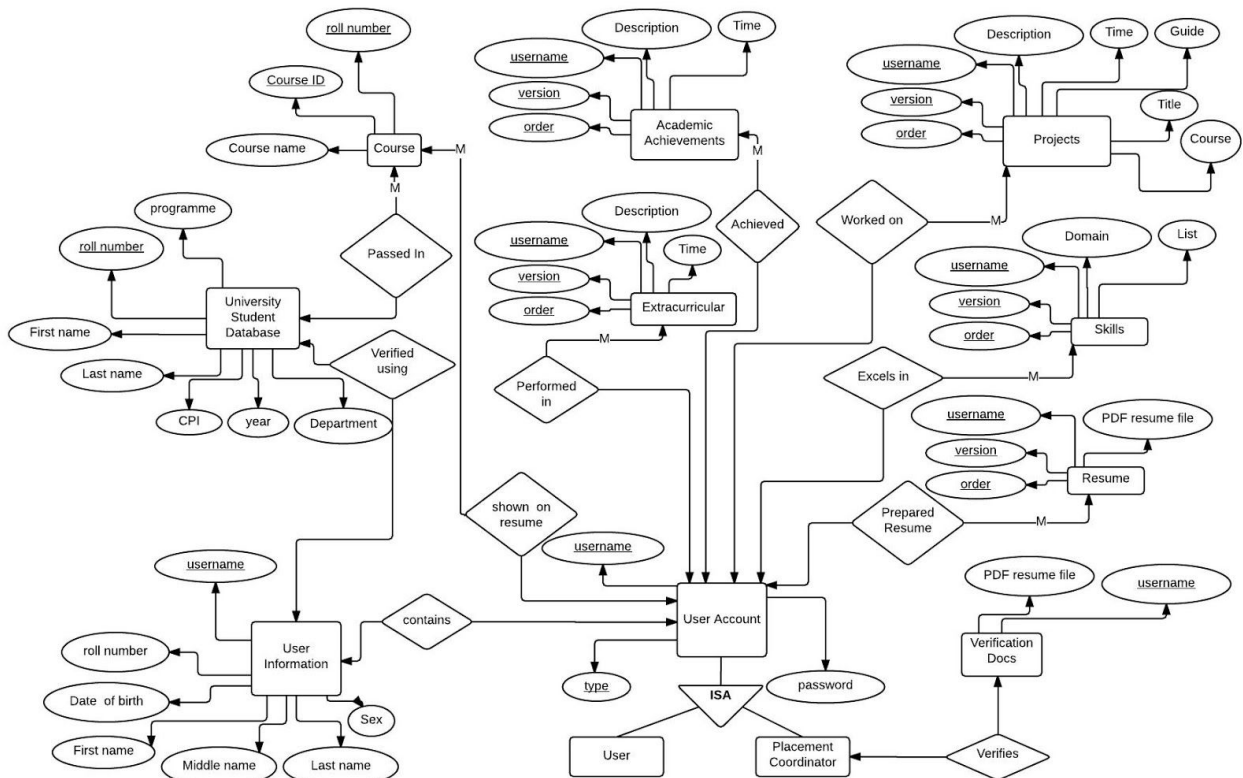


figure 1

---

## Normalization

Normalization is used to eliminate data redundancy in the database. We did the same in our design. It also helped us to ensure that data is stored in a logical way.

Current version of our database implements 3NF where we have removed extra dependencies which cause redundancy in the data.

## Tables

**Accounts :** This table stores username, password, type for students/Placement co-ordinators who currently have an account on the portal.

**User Information :** A table for fixed user informations which users will input during signup. It stores Name(first,middle,last), Roll number, DoB, Sex, etc.

**University Database :** This table stores information pertaining to university such as Roll Number, Name, CPI, Year, Department, Programme, etc. This information will be used for account verification and will be imported to Resume.

**Achievements :** A table to store Academic achievements corresponding to particular user.. This will be one-to-many mapping.

**Projects :** This table stores details related to projects done by users such as Guide, Course, Description, etc.

**Skills :** This table stores user's skills in various domains say, Computer languages, Tools, etc.

**Courses :** It has courses that a user wants to put on a resume. It will be verified from University Database.

**Extracurriculars :** A table to store Extracurricular achievements corresponding to particular user. This will be one-to-many mapping.

**Submitted Resumes :** This will contain the final version of resume submitted by particular user.

**Documents :** This stores documents submitted for verification.

## Indexing

Importing data from all the tables for a particular resume is the bottleneck of this application as there can be many versions. But data from all the tables is required for particular [username, version] at a time. Hence, we keep an index on username which makes these transactions very fast. University database (mainly courses) is expected to be much larger than other tables. Therefore we also keep an index on it. It will be a primary index using Roll no. as a key.

`CREATE INDEX index_ach ON Achievements (username, version, order);`

Similarly for other tables like Projects, Extracurriculars, etc.

`CREATE INDEX index_courses ON University_Courses (Roll No, Course ID);`

## Database Schema

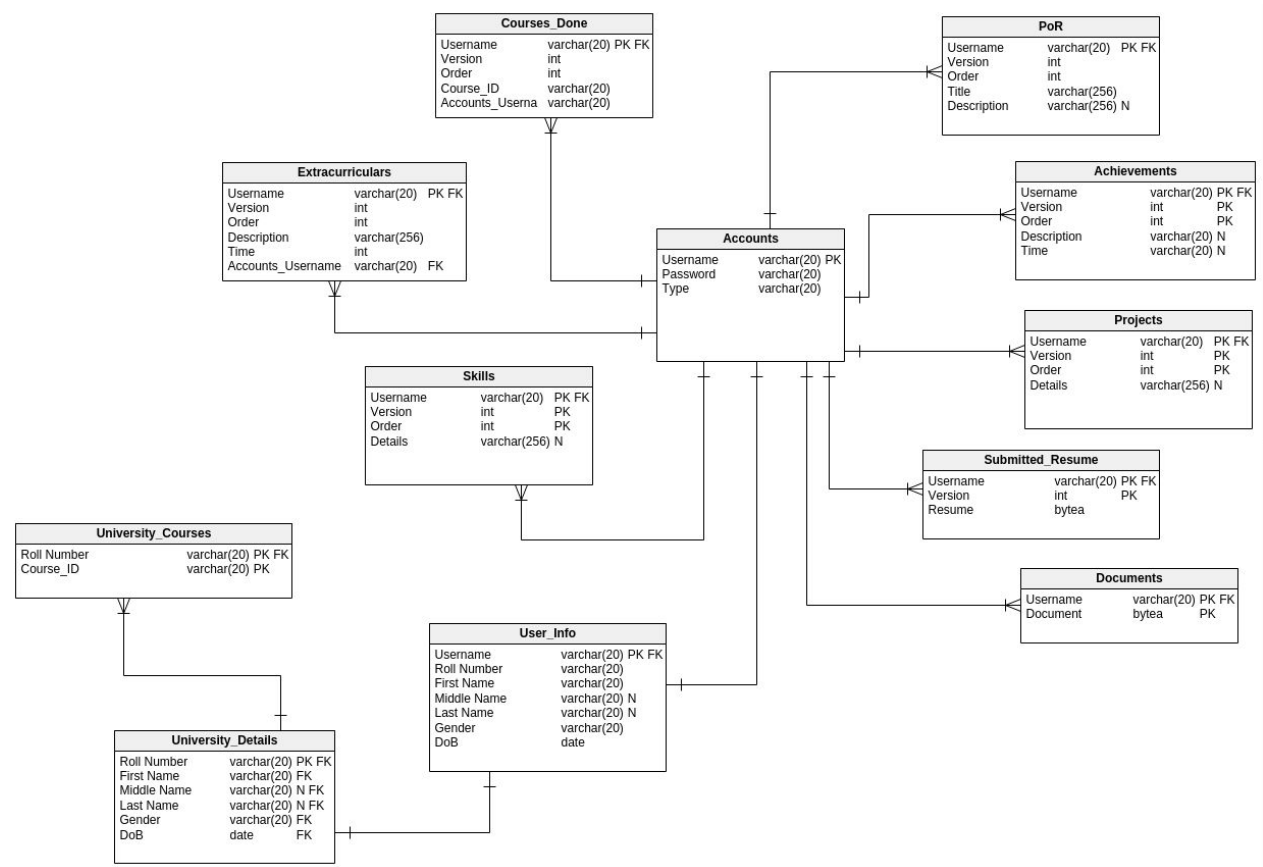


figure 2

\*\* PK: Primary Key

FK: Foreign Key

N: Nullable

---

## Implementation Details

### Classes :

1. Backend : This class implements the backend of our application. It provides functions to Handle file operations, Compile Resumes, Upload to database, Download from the database, etc.
2. LoginAccess : This class handles all the operations related to accounts. It allows user to create new or use existing account. It also implements the functions to connect to database.

### Servlets :

1. pdfViewer : When user compiles his data, this servlet takes username, version and passes it to function of Backend class to process data, generate the resume and render it on the screen.
2. pcViewer : This servlet passes information to Backend class and allows Placement Coordinator to view the resume when he/she selects the user for verification.
3. addRemark : This servlet handles the remark which are added by the placement coordinators for the user's version of resume.
4. deleteDocs : This allows the functionality for user to delete their documents from the Database.
5. deleteResume : This allows the user to delete their resume from the Database.
6. docViewer : This servlets downloads documents from servlet for user to view them.
7. fileUploader : Using this the application can handle file uploading from the users.
8. verifyRes/unverifyRes : This handles the resume verification related details.

### JSPs :

1. login.jsp: Users will be asked for username/password and will be redirected to his account. If either of their username or password is wrong they will be asked to go login window. Users can also go to signup window from here (See figure 3).
2. signup.jsp/signedin.jsp : User can create an account here. User will be asked to enter some of his/her information which will later be used in the resume and for account verification such as university roll number, name for verification and date of birth, gender etc for resume (see figure 4).
3. home.jsp : This will be a dashboard for the user where he/she can view various versions already submitted and can choose to edit them.
4. info.jsp : User can enter information which he/she wants to be appeared on their resume such as Achievements, Projects, Extracurriculars, etc (see figure 5).
5. pcMain.jsp : This opens a window for placement coordinator where he/she can view resumes uploaded by students and can also verify them. Coordinators can also see inform users for their requirements for verification.

6. [docUpload.jsp](#) : Users can upload documents for verification on this window, which can be later verified by, placement coordinators. User can also delete older or not of use documents here.

## External Packages Used

**Jars** : [commons-fileupload-1.3.1.jar](#) - for File uploading servlet

[commons-io-2.4.jar](#) - helper to above jar file

[servlet-api-3.0.jar](#) - Advanced version of servlet-api.jar

**CSS** : Twitter Bootstrap

## How to run

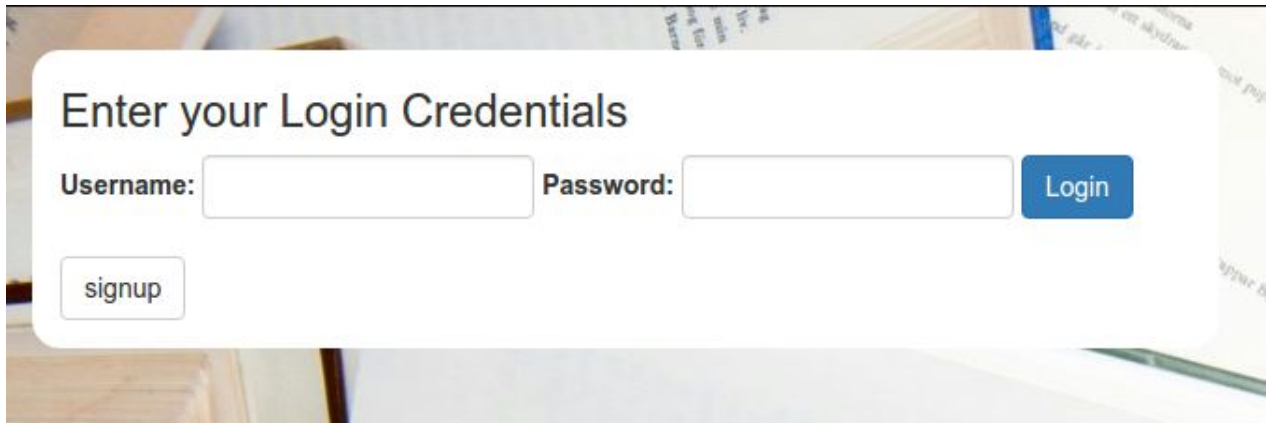
1. Keep the resumeG directory at some location and in Backend.java change “String dirc” to the location of resumeG folder.
2. Build path to above given jar files which are present in WEB-INF/lib
3. Active internet connection is required for bootstrap css to work.
4. Run `sampledata.sql` on the database server.
5. Change the database server specification in the “LoginAccess.java” class.
6. Start the tomcat server.

## Interface Design

**Signup :**

figure 3

## Login :

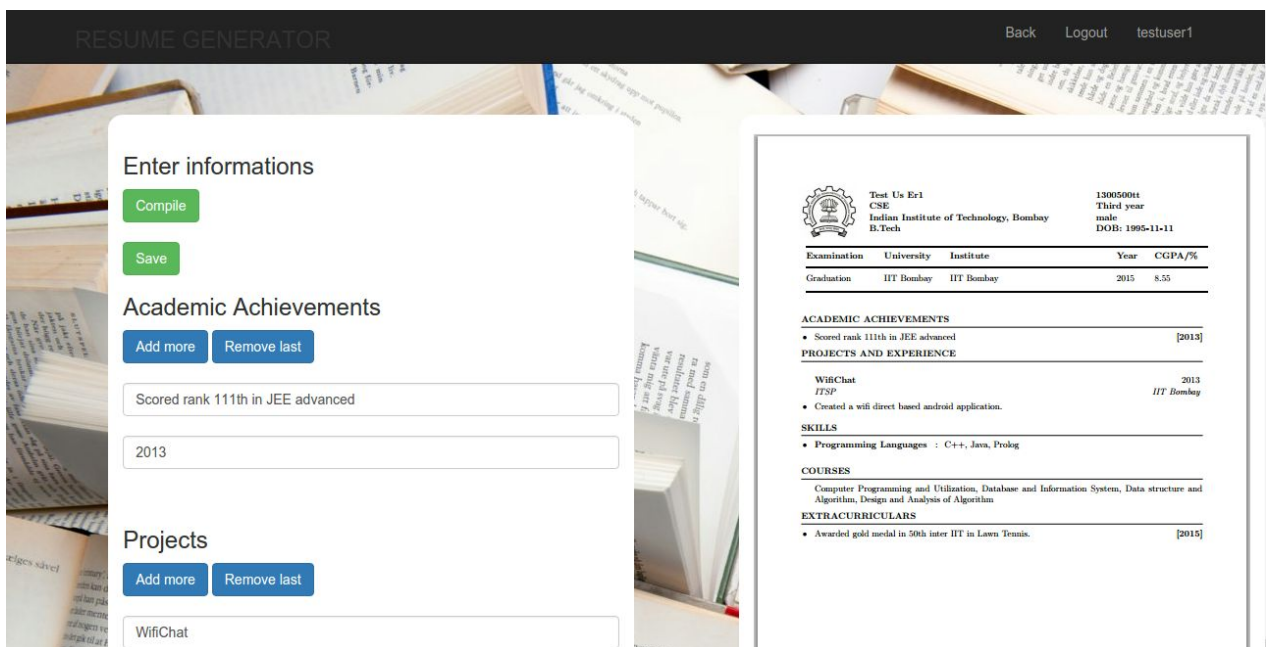


The screenshot shows a login form with the following elements:

- Title: Enter your Login Credentials
- Username:
- Password:
- Login button (blue)
- signup button (white)

figure 4

## Create Resume :



The screenshot shows a resume generator interface with the following sections:

- Header: RESUME GENERATOR, Back, Logout, testuser1
- Enter informations: Compile, Save
- Academic Achievements: Add more, Remove last, Scored rank 111th in JEE advanced, 2013
- Projects: Add more, Remove last, WifiChat
- Preview of a resume:

**Test Us Er1**  
CSE  
Indian Institute of Technology, Bombay  
B.Tech

13005004  
Third year  
male  
DOB: 1995-11-11

Examination	University	Institute	Year	CGPA/%
Graduation	IIT Bombay	IIT Bombay	2015	8.55

**ACADEMIC ACHIEVEMENTS**

- Scored rank 111th in JEE advanced [2013]

**PROJECTS AND EXPERIENCE**

**WifiChat** 2013  
IIT Bombay

- Created a wifi direct based android application.

**SKILLS**

- Programming Languages : C++, Java, Prolog

**COURSES**

Computer Programming and Utilization, Database and Information System, Data structure and Algorithm, Design and Analysis of Algorithm.

**EXTRACURRICULARS**

- Awarded gold medal in 50th inter IIT in Lawn Tennis. [2015]

figure 5



## Document uploading:

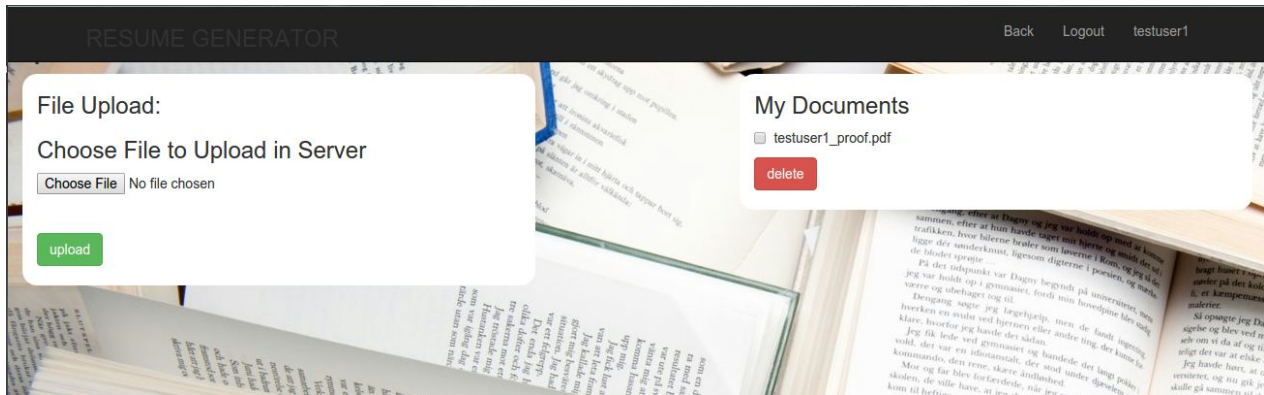


figure 6

## Home of user:

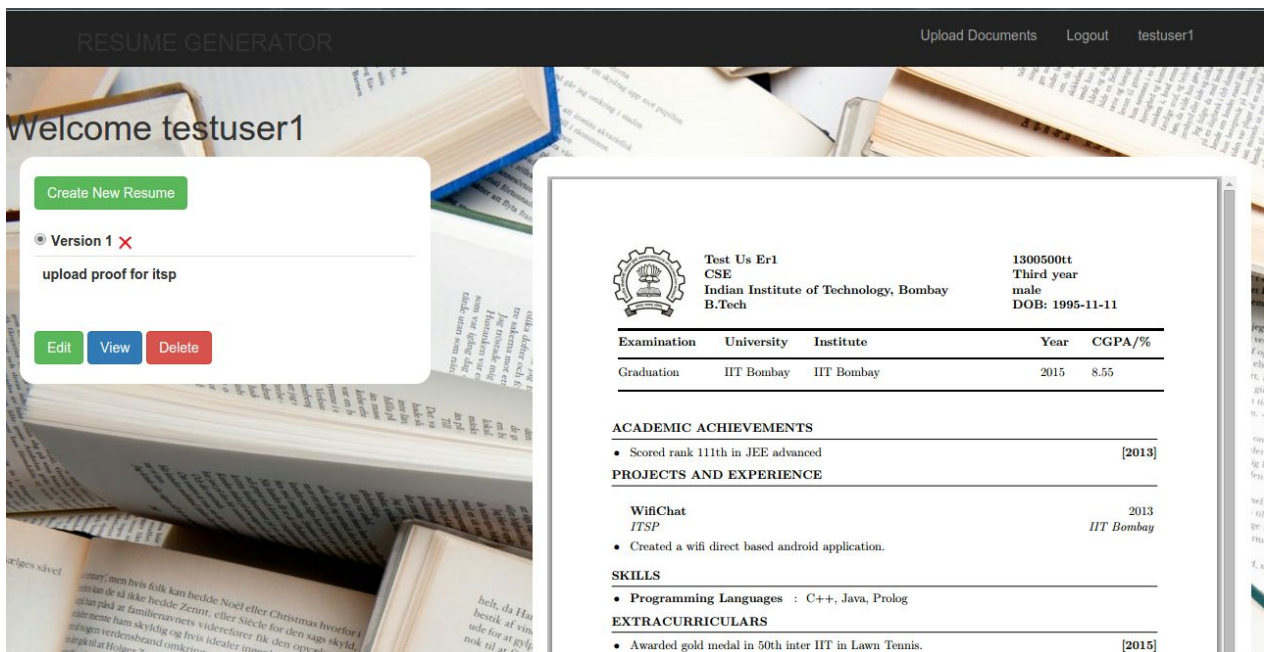


figure 6

## Window for placement coordinator:

The screenshot shows the 'RESUME GENERATOR' interface. On the left, there is a 'List of Resume to verify' section with buttons for 'testuser1 Version 1', 'a Version 2', and 'a Version 1'. Below this, 'Documents for testuser1' includes 'testuser1\_proof.pdf' with 'Verify' and 'Revert Verification' buttons, and a 'Reload status' button. A 'remarks' input field and a 'Submit' button are also present. On the right, a detailed resume for 'Test Us Er1' is displayed, including personal information, a table of examinations, and sections for 'ACADEMIC ACHIEVEMENTS', 'PROJECTS AND EXPERIENCE', 'SKILLS', and 'EXTRACURRICULARS'.

Examination	University	Institute	Year	CGPA/%
Graduation	IIT Bombay	IIT Bombay	2015	8.55

**ACADEMIC ACHIEVEMENTS**

- Scored rank 111th in JEE advanced [2013]

**PROJECTS AND EXPERIENCE**

WiFiChat  
ITSP IIT Bombay 2013

- Created a wifi direct based android application.

**SKILLS**

- Programming Languages : C++, Java, Prolog

**EXTRACURRICULARS**

- Awarded gold medal in 50th inter IIT in Lawn Tennis. [2015]

figure 7

## Modal view for document viewing:

The screenshot shows the 'testuser1\_proof.pdf' modal view. The modal title is 'testuser1\_proof.pdf'. The document content is a 'Proof for Equivalency' with the following sections:

**1 Properties**

Here we assume the MDP is a 2 action MDP i.e. the action at any state in any policy is either 0 or 1. For any policy  $p_i$  of all the states belonging to it, the states, switching will improve the policy is called Improvement Set of Policy  $p_i$ . It is represented by  $IS(p_i)$ . Since we are using Howard's Policy Iteration, all the states in  $IS(p_i)$  are switched. The improvement set can also be represented as a subset below-mentioned with it,  $P_i$  element as  $1 \leq i \leq n$ ,  $g \in IS(p_i)$ .

Any new policy we get by switching only in  $IS(p_i)$  (not necessarily Howard's switch) dominates current policy  $p_i$ . We are calling this set Improved Policy Set, represented by  $IPS(p_i)$ . Also any policy we get by switching only in  $S \setminus IS(p_i)$  is dominated by  $p_i$  itself, where  $S$  is the set of all states. We are calling the set Deproved Policy Set, represented by  $DPS(p_i)$ .

**1.1 Subset Condition**

Let  $P = \langle p_1, p_2, p_3, \dots, p_m \rangle$  be the sequence of policies we get from policy iteration. So, for any  $i, j \in \{1, 2, \dots, m\}$  if  $i < j$  then,  $IS(p_i) \not\subseteq IS(p_j)$ . This can be written as:

$$\forall i < j (IS(p_i) \not\subseteq IS(p_j))$$

**1.2 Intersection Condition**

Let  $P = \langle p_1, p_2, p_3, \dots, p_m \rangle$  be the sequence of policies we get from policy iteration. So, for any  $i, j \in \{1, 2, \dots, m\}$  if  $i < j$  then,  $DPS(p_i)$ , should not have any element from  $IPS(p_j)$ . This can be written as:

$$\forall i < j (DPS(p_i) \cap IPS(p_j) = \emptyset)$$

**1.3 OR Matrix Condition**

figure 8