Assessing the Impact of Re-engineering on Software Security

Presented By:
Team 7
Shreya Reddy Gade
Yogesh Kalapala
Sivathejeswara Reddy Komma
Nalini Pasupuleti

UTA

Coding Assignment 1&2: MavAppoint & SAMS

Reused Requirements:

- Hashed Passwords : Stored the passwords using the hashing method in the database.
- Calendar : Appointments made reflect on the calendar.
- Edit Appointments : Ability to cancel or reschedule appointments.

Modified Requirements:

- Register Student: Create a one-time account and use it to schedule appointment with advisors.
- Email : A temporary password will be sent to the particular email when first time logged into the account and send a reminder email to the student email.

Created Database schema by getting the requirements from the given code.

Coding Assignment 2: Samsdb

Requirements:

Requirement ID	Requirement Statement	Category	Priority
R1	The system shall provide a function to login to the application. A login page for admin and student users, with appropriate authentication and authorization mechanisms, to access the admin dashboard and other features related to admin.	M	1
R2	The system shall provide function to reset the password	N	2
R3	The administrator is responsible for following activities on user profile:		
R3.1	The admin shall have an option to create a user	M	1
R3.2	The admin shall have option to remove a user	M	4
R3.3	The admin shall have an option to update the user details	M	3
R4	The system shall validate the email address, and upon validation, shall send a temporary password to the student's UTA email address	N	1
R5	The system shall implement ReCAPTCHA for the user and admin login pages to prevent automated bot attacks	N	2
R6	The system shall implement Hashed password to enhance the security, store the passwords using the hashing method in the database	N	2

M- Modified
N- New
R- Reused

1- Highest priority

Requirements Contd…

R7	The administrator is responsible for following on Program management:		
R7.1	The admin shall add a program into the system	R	1
R7.2	The admin shall search the program from the database	R	3
R7.3	The admin shall edit the program details	R	3
R7.4	The admin shall delete the program from the database	R	4
R7.5	The admin shall upload program file into the system	M	3
R8	The system shall provide the users with following functions having a role of		
R8.1	Student to provide Feedback on advising session	N	5
R8.2	Admin to view the feedback provided by all the users	N	5
R9	The system shall provide function to logout of the application	R	4

Use Cases:

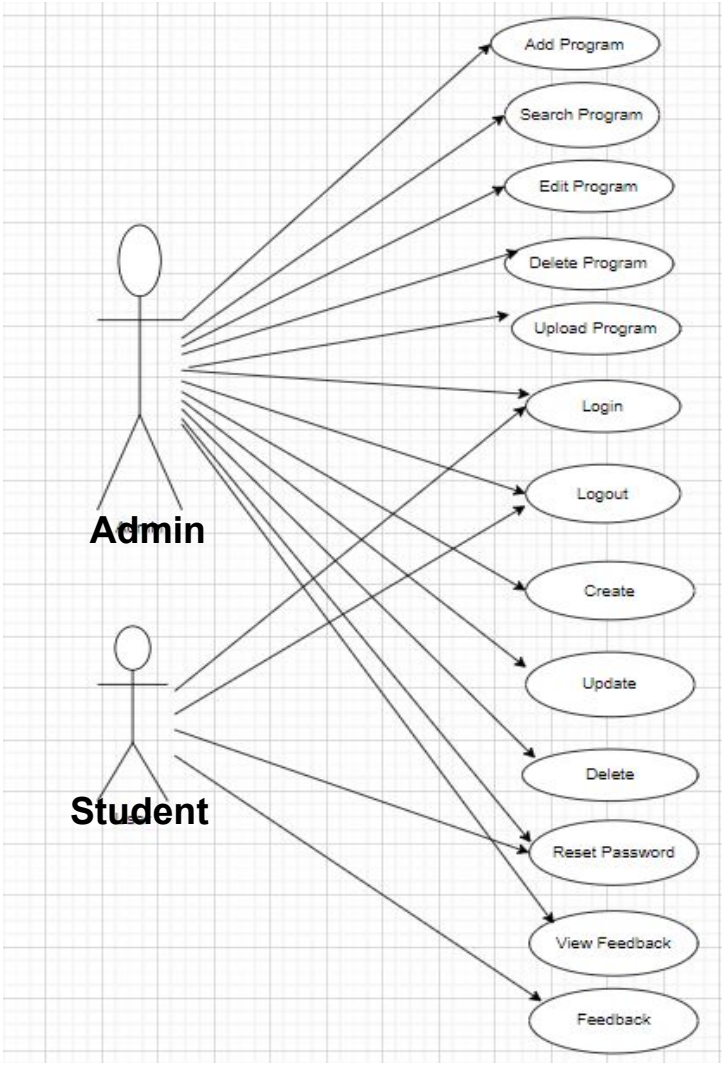
Use Case #	Use Case Name
UC1	Login
UC2	Reset Password
UC3	Add Program
UC4	Search Program
UC5	Edit Program
UC6	Delete Program
UC7	Upload Program
UC8	View Feedback
UC9	Create User
UC10	Update User
UC11	Delete User
UC12	Feedback
UC13	Logout

Requirements To Use Case Traceability Matrix:

Requirement ID	Priority Weight	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8	UC9	UC10	UC11	UC12	UC13
R1	1	X												
R2	2		X											
R3														
R3.1	1									X				
R3.2	4											X		
R3.3	3										X			
R4	1	X												
R5	2	X												
R6	2	X												
R7														
R7.1	1			X										
R7.2	3				X									
R7.3	3					X								
R7.4	4						X							
R7.5	3							X						
R8														
R8.1	5								X					
R8.2	5												X	
R9	4													X
	Score	6	2	1	3	3	4	3	5	1	3	4	5	4

1- Highest priority
5- Lowest priority

Use Case Diagram:

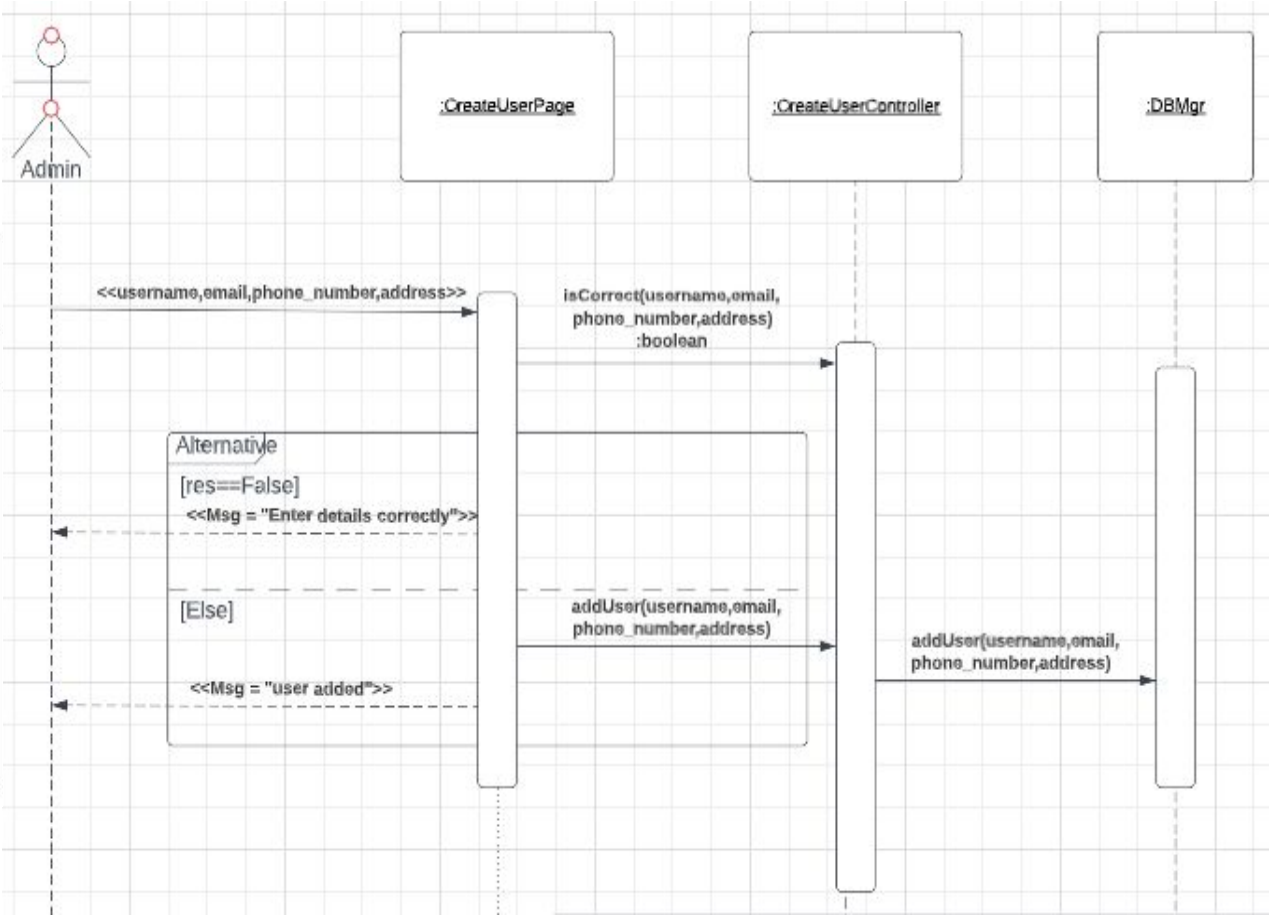


Users: Admin, Student

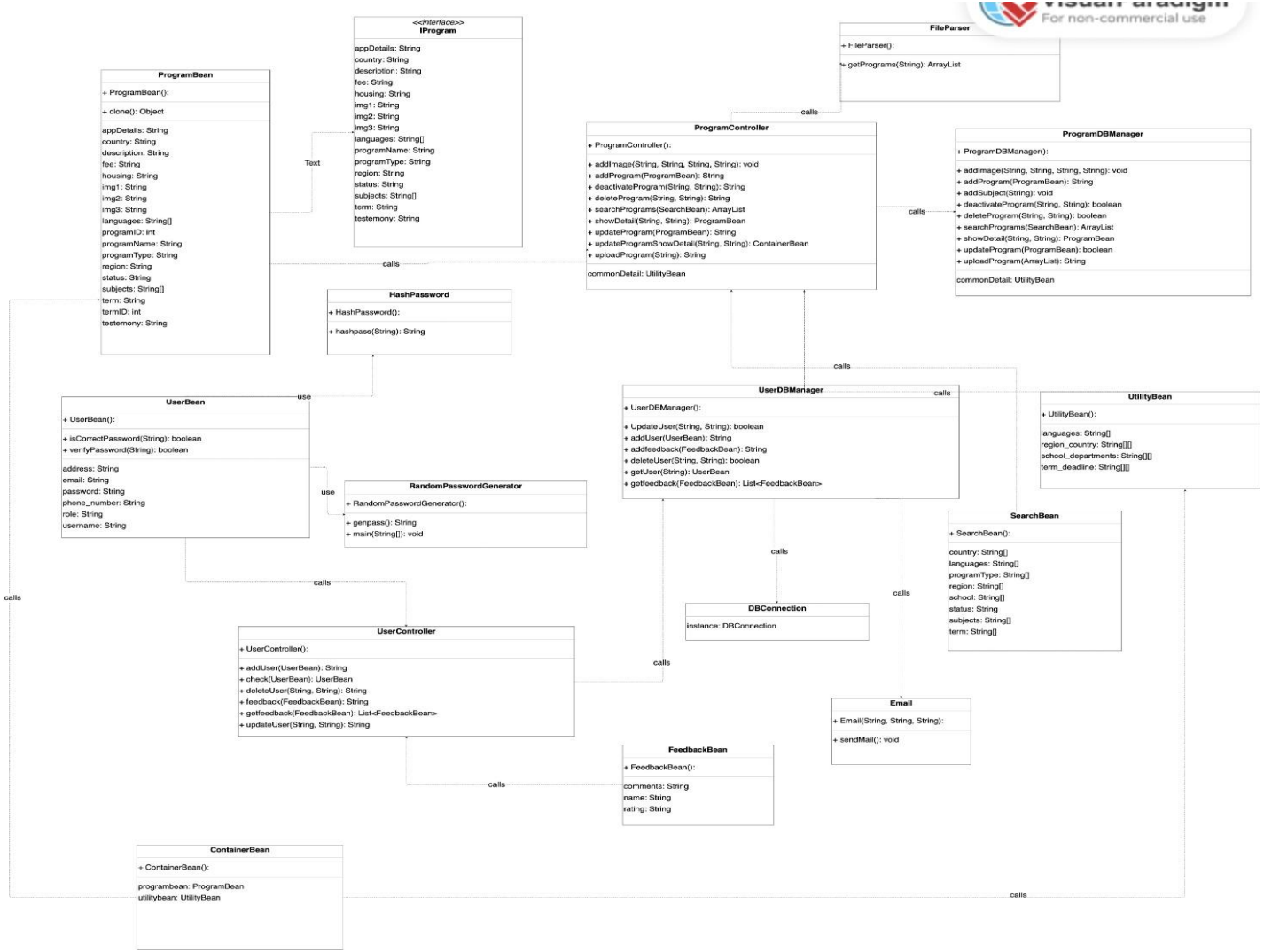
Sequence Diagram:

Create User:

Precondition: This use case assumes that user is in a create user page and wants to add user.	
Actor: Admin	System: Samsdb
1. TUCBW the admin fills the form and clicks on the 'Add User' button. 4. TUCEW the admin getting a success message and the user being added to database.	0. System will display the create user page created. And display the form to fill in the data. 2. *System Add the users in the group and displays success message.



Domain Class Diagram:



Paper Title: Assessing the Impact of Re-engineering on Software Security
Authors: Shreya Reddy Gade, Yogesh Kalapala, Sivethejeswara Reddy Komma, Nalini Pasupuleti
Year: 2023

Main Idea:

- Software reengineering is essential to keep up with rapidly evolving technology, user expectations, and business objectives.
- Reengineering, however, may also result in the introduction of fresh security holes that may harm people and software systems.
- During the reengineering process, security safeguards must be put in place by developers, users, and organizations to protect against security threats.
- By following best practices and suggestions for lowering security risks, the software reengineering process can be made safer and more resilient.
- Case studies have been conducted to support our assertions.
- Software systems can be badly harmed by a number of security hazards, including viruses, malware, hacking, and cyberattacks. These risks can also result in data breaches, endanger user privacy, and compromise system security.
- making sure that software systems are secure and resistant to potential security attacks is crucial.

Paper Title: Assessing the Impact of Re-engineering on Software Security
Authors: Shreya Reddy Gade, Yogesh Kalapala, Sivethejeswara Reddy Komma, Nalini Pasupuleti
Year: 2023

Motivation:

Software Re-engineering and Security

- In order to mitigate these risks, it is crucial to review best practices and guidelines for limiting the effects of software reengineering on security. Software re-engineering entails changing existing code to increase usability, stability, and the ability to be maintained.
- It can aid companies in adapting to new demands and maintaining their competitiveness.
- However, re-engineering could also result in the creation of fresh security holes.
- Software security involves creating, constructing, and testing secure software.
- It tries to reduce software flaws that raise the risk of being vulnerable to attack.
- During the early stages of the development cycle, developers can identify potential vulnerabilities by conducting unbiased risk analyses and testing on all software components.
- Language-based weaknesses like SQL injection attacks, cross-site scripting (XSS) attacks, and buffer overflow attacks can be avoided by using secure coding practices like input validation and output encoding.
- Setting security as a top priority early on can help to lower risks and guarantee dependable, effective, and affordable software systems.
- hazards to security throughout the reengineering process.

Paper Title: Assessing the Impact of Re-engineering on Software Security
Authors: Shreya Reddy Gade, Yogesh Kalapala, Sivethejeswara Reddy Komma, Nalini Pasupuleti
Year: 2023

Reengineering Methodology

- Reverse Engineering: Utilize models to create abstract representations.
- Security Analysis: Identify vulnerabilities and their potential impact
- Model Transformation: Update models with security enhancements and best practices
- Forward Engineering: Generate new code adhering to desired security properties
- Validation and Verification: Conduct rigorous security testing to validate and verify effectiveness of enhancements

Tools

- Reverse Engineering Tools: extract information about the existing software system
- Model Transformation Tools: transform extracted information into models for analysis
- Model Analysis Tools: analyze the transformed models to identify issues
- Code Refactoring Tools: modify existing codebase to adhere to new design
- Automated Testing Tools: perform functional, performance, and security testing of re-engineered system.

Paper Title: Assessing the Impact of Re-engineering on Software Security
Authors: Shreya Reddy Gade, Yogesh Kalapala, Sivethejeswar Reddy Komma, Nalini Pasupuleti
Year: 2023

Result:

- **Goals and objectives**: Different methodologies have different goals, e.g., Model-Driven Re-engineering (MDRE) focuses on identifying vulnerabilities at the model level, while Architecture-Centric Re-engineering (ACRE) addresses security vulnerabilities arising from architectural flaws.
- **Techniques and tools**: Re-engineering methodologies employ various techniques and tools for security improvement, such as security patterns, static analysis tools, and penetration testing.
- **Security improvements**: Results indicate that different methodologies achieve varying degrees of security improvements depending on their goals, techniques, and tools, and the system's context.

Limitations:

- **Complexity of software systems**: During the re-engineering process, it might be challenging to find and fix all potential security flaws due to the complexity of current software systems.
- **Limited security expertise**: The efficient application of re-engineering processes and procedures is frequently hampered by the lack of security expertise among software developers and engineers.
- **Resource limitations**: It can be challenging to allocate sufficient resources for improvements to security because the re-engineering process can be time- and resource-intensive.

Weakness:

- Older and poorly maintained software is vulnerable to known bugs and vulnerabilities.
- Incompatibility with the latest security measures and updates puts sensitive data at risk.
- Reengineering allows for necessary system reorganization or redesign, but can also introduce vulnerabilities.
- Following reengineering and security models is crucial to mitigate security risks.
- Various models for software reengineering and security can complement each other to enhance system security performance.

UTA