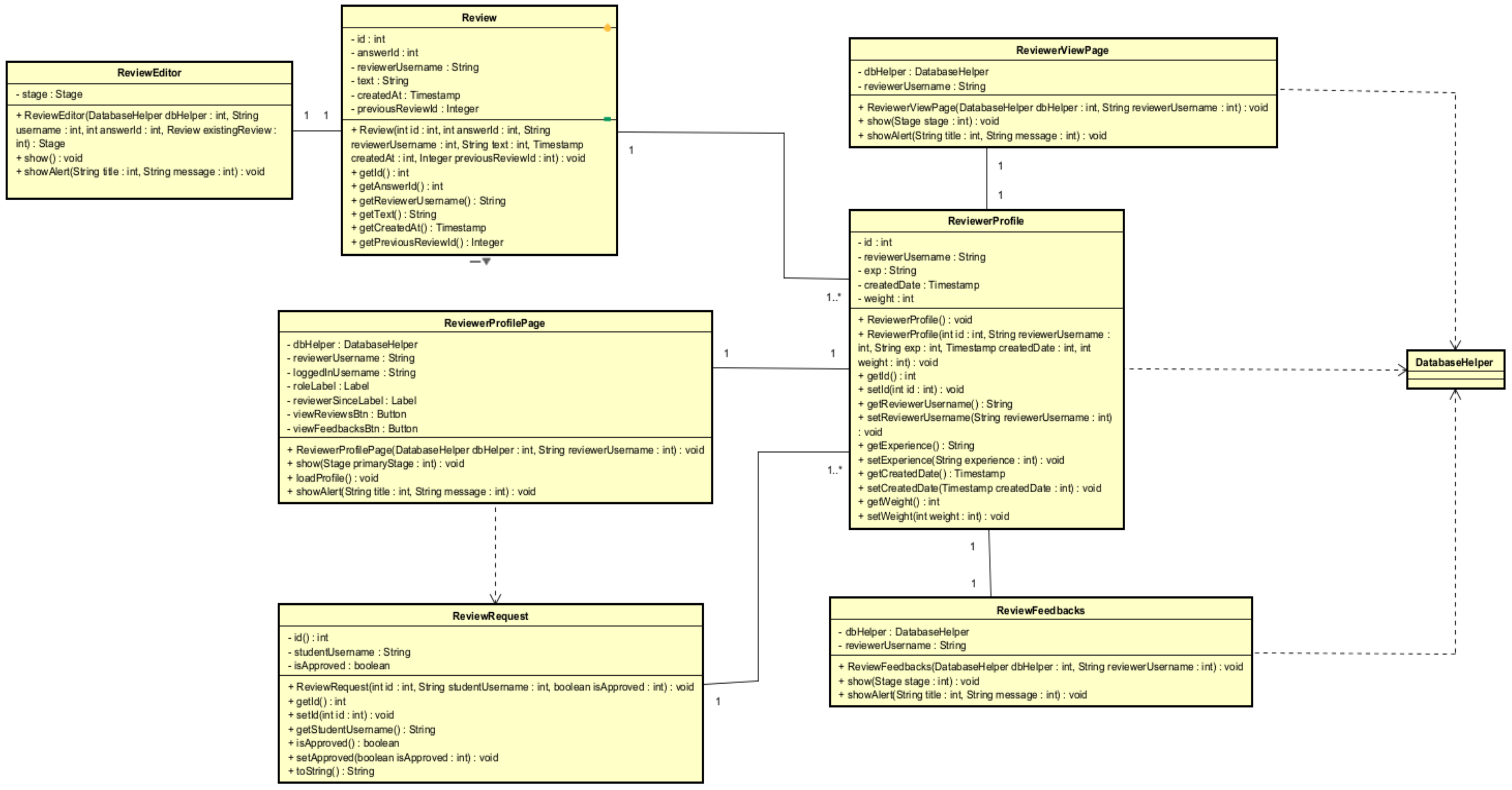
Architecture and Design Document

Introduction

This document outlines the architecture and design of the software implementation, detailing both the existing code from TP2 and the new additions made in TP3. By ensuring a cohesive structure between code, design, and documentation, this document aims to provide both clarity and transparency for anyone reviewing or working with the system in the future. It is separated into two sections, Phase 3 Additional Code UML and Phase 2 UML

Phase 3 Additional Code UML


ReviewEditor
- stage : Stage
+ ReviewEditor(DatabaseHelper dbHelper : int, String username : int, int answerId : int, Review existingReview : int) : Stage
+ show() : void
+ showAlert(String title : int, String message : int) : void
Review
- id : int
- answerId : int
- reviewerUsername : String
- text : String
- createdAt : Timestamp
- previousReviewId : Integer
+ Review(int id : int, int answerId : int, String reviewerUsername : int, String text : int, Timestamp createdAt : int, Integer previousReviewId : int) : void
+ getId() : int
+ getAnswerId() : int
+ getReviewerUsername() : String
+ getText() : String
+ getCreatedAt() : Timestamp
+ getPreviousReviewId() : Integer
ReviewerViewPage
- dbHelper : DatabaseHelper
- reviewerUsername : String
+ ReviewerViewPage(DatabaseHelper dbHelper : int, String reviewerUsername : int) : void
+ show(Stage stage : int) : void
+ showAlert(String title : int, String message : int) : void
ReviewerProfile
- id : int
- reviewerUsername : String
- exp : String
- createdDate : Timestamp
- weight : int
+ ReviewerProfile() : void
+ ReviewerProfile(int id : int, String reviewerUsername : int, String exp : int, Timestamp createdDate : int, int weight : int) : void
+ getId() : int
+ setId(int id : int) : void
+ getReviewerUsername() : String
+ setReviewerUsername(String reviewerUsername : int) : void
+ getExperience() : String
+ setExperience(String experience : int) : void
+ getCreatedDate() : Timestamp
+ setCreatedDate(Timestamp createdDate : int) : void
+ getWeight() : int
+ setWeight(int weight : int) : void
ReviewerProfilePage
- dbHelper : DatabaseHelper
- reviewerUsername : String
- loggedInUsername : String
- roleLabel : Label
- reviewerSinceLabel : Label
- viewReviewsBtn : Button
- viewFeedbacksBtn : Button
+ ReviewerProfilePage(DatabaseHelper dbHelper : int, String reviewerUsername : int) : void
+ show(Stage primaryStage : int) : void
+ loadProfile() : void
+ showAlert(String title : int, String message : int) : void
DatabaseHelper
ReviewRequest
- id() : int
- studentUsername : String
- isApproved : boolean
+ ReviewRequest(int id : int, String studentUsername : int, boolean isApproved : int) : void
+ getId() : int
+ setId(int id : int) : void
+ getStudentUsername() : String
+ isApproved() : boolean
+ setApproved(boolean isApproved : int) : void
+ toString() : String
ReviewFeedbacks
- dbHelper : DatabaseHelper
- reviewerUsername : String
+ ReviewFeedbacks(DatabaseHelper dbHelper : int, String reviewerUsername : int) : void
+ show(Stage stage : int) : void
+ showAlert(String title : int, String message : int) : void
1
1
1
1..*
1
1
1
1
1..*
1
1
1
1

ReviewerProfilePage → ReviewFeedbacks (Association)

- **Explanation**: The ReviewerProfilePage class contains a button labeled viewFeedbacksBtn, which, when triggered, creates an instance of the ReviewFeedbacks class and calls its show() method to display the feedback information.

- **Type**: One-to-one association.

- **Reasoning**: The ReviewerProfilePage directly instantiated ReviewFeedbacks when the user opts to view feedback, establishing a clear one-to-one relationship between the two components. This interaction is designed for seamless navigation from the profile page to the feedback page.

ReviewerProfilePage → ReviewerViewPage (Association)

- **Explanation**: Similarly to the previous relationship, ReviewerProfilePage contains another button, viewReviewsBtn, which creates an instance of ReviewerViewPage and invokes its show() method to display the list of reviews.

- **Type**: One-to-one association.

- **Reasoning**: This is a one-to-one relationship because each click of the viewReviewsBtn results in the creation of a single ReviewerViewPage instance. The navigation is intended to provide an individual review view, ensuring that each page is unique to the instance of the ReviewerViewPage.

ReviewerProfilePage → DatabaseHelper (Dependency)

- **Explanation**: The ReviewerProfilePage requires a DatabaseHelper instance to interact with the database and retrieve reviewer details (e.g., the reviewer's profile data). The DatabaseHelper is used to query the database and provide the necessary information to the profile page.

- **Type**: Dependency.

- **Reasoning**: ReviewerProfilePage depends on the DatabaseHelper for its database interactions. This dependency is required to ensure that reviewer data is retrieved dynamically from the database. By using a dependency injection approach (presumably through the constructor or setter methods), the code decouples the profile page from the database implementation, allowing for better modularity and testability.

ReviewerViewPage → DatabaseHelper (Dependency)

- **Explanation**: Similar to ReviewerProfilePage, the ReviewerViewPage requires DatabaseHelper to retrieve the list of reviews associated with a particular reviewer. It uses the database helper to fetch review data that is then presented to the user.

- **Type**: Dependency.

- **Reasoning**: This dependency ensures that the ReviewerViewPage can dynamically load review data as needed, which is essential for providing up-to-date information. By abstracting the database interactions behind the DatabaseHelper, the design promotes separation of concerns and easier maintenance.

ReviewFeedbacks → DatabaseHelper (Dependency)

- **Explanation**: The ReviewFeedbacks class fetches reviewer feedback from the database, leveraging DatabaseHelper to query and retrieve the necessary feedback data.

- **Type**: Dependency.

- **Reasoning**: ReviewFeedbacks relies on DatabaseHelper to access feedback data stored in the database. This dependency facilitates centralized data access and ensures that feedback retrieval is consistent with other data retrieval processes in the system.

ReviewerProfilePage → ReviewRequest (Indirect Association)

- **Explanation**: The ReviewerProfilePage interacts indirectly with the ReviewRequest class via the DatabaseHelper. Specifically, when dbHelper.getReviewerProfile(reviewerUsername) is called, it may invoke methods that interact with the ReviewRequest class, thereby establishing an indirect relationship.

- **Type**: Indirect association.

- **Reasoning**: While the ReviewerProfilePage does not directly interact with ReviewRequest, the data it retrieves may reference review requests (such as retrieving a list of reviews linked to a specific reviewer). This indirect interaction is modeled to separate concerns, ensuring that profile retrieval and review request processing are handled in distinct components.

ReviewRequest → ReviewerProfile (Association)

- **Explanation**: The ReviewRequest class contains a studentUsername field, which is associated with a specific reviewer profile. Each ReviewRequest could be linked to one or more reviewers, depending on how the system is designed to manage review assignments.
- **Type**: One-to-one
- **Reasoning**: A single ReviewRequest may be associated with multiple reviewers (if the request involves a group of reviewers), or it may be linked to just one reviewer. This design allows flexibility in how reviews are assigned and managed while maintaining a clear connection between requests and the reviewers involved.

ReviewerProfile → ReviewerProfilePage (Association)

- **Explanation**: The ReviewerProfile class is used to store and manage the reviewer's personal details. The ReviewerProfilePage loads this information to display the profile data to the user.
- **Type**: One-to-one association.
- **Reasoning**: Each reviewer profile corresponds to a specific page displaying that profile's data. This direct, one-to-one relationship simplifies the user interface design and ensures that a reviewer can only have one profile displayed at a time on the ReviewerProfilePage.

ReviewEditor → Review (Association)

- **Explanation**: The ReviewEditor class is responsible for editing existing reviews. To do so, it needs access to an instance of the Review class, which it modifies based on user input.
- **Type**: One-to-one association.
- **Reasoning**: The ReviewEditor is designed to edit a single Review at a time, so a one-to-one association is appropriate. This keeps the interaction straightforward and prevents the need for complex management of multiple reviews during editing.

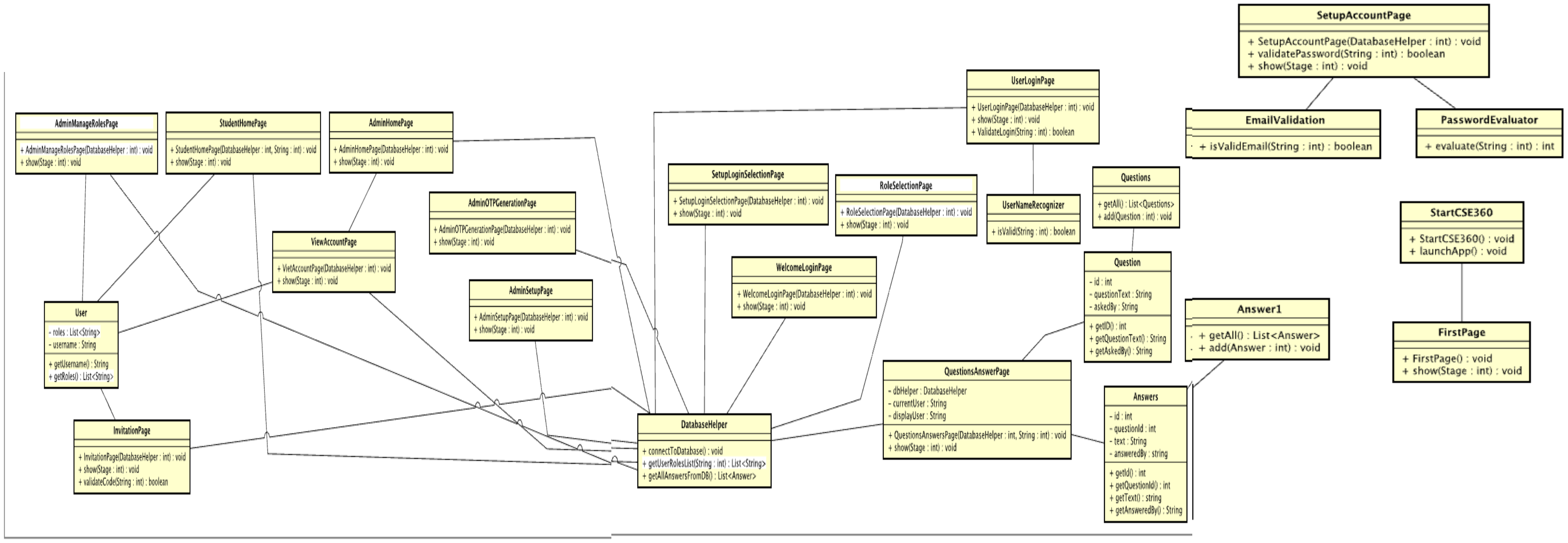
Review → ReviewerProfile (Association)

- **Explanation**: Each Review belongs to a specific reviewer, identified by the reviewerUsername field. This establishes a connection between a review and the reviewer who authored it.

- **Type**: One-to-many association.

- **Reasoning**: A single reviewer can write multiple reviews, so the relationship between Review and ReviewerProfile is one-to-many. This design supports the possibility of a reviewer having a series of reviews across various subjects or projects.

Phase 2 UML (Split up because it's too wide)


SetupAccountPage
+ SetupAccountPage(DatabaseHelper : int) : void
+ validatePassword(String : int) : boolean
+ show(Stage : int) : void
UserLoginPage
+ UserLoginPage(DatabaseHelper : int) : void
+ show(Stage : int) : void
+ ValidateLogin(String : int) : boolean
EmailValidation
+ isValidEmail(String : int) : boolean
PasswordEvaluator
+ evaluate(String : int) : int
AdminManageRolesPage
+ AdminManageRolesPage(DatabaseHelper : int) : void
+ show(Stage : int) : void
StudentHomePage
+ StudentHomePage(DatabaseHelper : int, String : int) : void
+ show(Stage : int) : void
AdminHomePage
+ AdminHomePage(DatabaseHelper : int) : void
+ show(Stage : int) : void
SetupLoginSelectionPage
+ SetupLoginSelectionPage(DatabaseHelper : int) : void
+ show(Stage : int) : void
RoleSelectionPage
+ RoleSelectionPage(DatabaseHelper : int) : void
+ show(Stage : int) : void
UserNameRecognizer
+ isValid(String : int) : boolean
Questions
+ getAll() : List<Questions>
+ add(Question : int) : void
AdminOTPGenerationPage
+ AdminOTPGenerationPage(DatabaseHelper : int) : void
+ show(Stage : int) : void
ViewAccountPage
+ VietAccountPage(DatabaseHelper : int) : void
+ show(Stage : int) : void
StartCSE360
+ StartCSE360() : void
+ launchApp() : void
Question
- id : int
- questionText : String
- askedBy : String
+ getID() : int
+ getQuestionText() : String
+ getAskedBy() : String
WelcomeLoginPage
+ WelcomeLoginPage(DatabaseHelper : int) : void
+ show(Stage : int) : void
AdminSetupPage
+ AdminSetupPage(DatabaseHelper : int) : void
+ show(Stage : int) : void
User
- roles : List<String>
- username : String
+ getUsername() : String
+ getRoles() : List<String>
Answer1
+ getAll() : List<Answer>
+ add(Answer : int) : void
FirstPage
+ FirstPage() : void
+ show(Stage : int) : void
QuestionsAnswerPage
- dbHelper : DatabaseHelper
- currentUser : String
- displayUser : String
+ QuestionsAnswersPage(DatabaseHelper : int, String : int) : void
+ show(Stage : int) : void
Answers
- id : int
- questionId : int
- text : String
- answeredBy : string
+ getId() : int
+ getQuestionId() : int
+ getText() : string
+ getAnsweredBy() : String
InvitationPage
+ InvitationPage(DatabaseHelper : int) : void
+ show(Stage : int) : void
+ validateCode(String : int) : boolean
DatabaseHelper
+ connectToDatabase() : void
+ getUserRolesList(String : int) : List<String>
+ getAllAnswersFromDB() : List<Answer>