System Design

EECS 3311
CodeEsc

Project Overview

The CodeESC project is an interactive escape room puzzle game designed to challenge players through various levels with engaging gameplay, a leaderboard system, and other features. The application will be developed using Java.

CRC Cards

Class Name: CodeEscMenu	
Parent Class: JFrame **Subclasses:**	
Responsibilities: -Initialize and set up the main menu window for the game. -Manage layout by adding background, button, and help panels. -Load the background image from resources and configure window settings.	**Collaborators**: -BackgroundPanel -ButtonPanel -HelpPanel

Class Name: ButtonPanel	
Parent Class: JPanel **Subclasses:**	
Responsibilities: -Create and style "Play", "Settings", and "Leaderboard" buttons. -Configure button behaviour, including navigation to other game scenes in the future.	**Collaborators**: -CodeEscMenu

Class Name: BackgroundPanel	
Parent Class: JPanel **Subclasses:**	
Responsibilities: -Display the background image for the main menu.	**Collaborators**: -CodeEscMenu

Class Name: HelpPanel	
Parent Class: JPanel **Subclasses:**	
Responsibilities: -Display a "Help" button -Show a help dialog with relevant information when clicked.	**Collaborators**: -CodeEscMenu

Class name: LoginForm	
Parent Class (if any): None **Subclasses (if any):** None	
Responsibilities: ● The main responsibility of the Login Form is to provide a UI where a user can login and sign-up into their respective accounts. ● To have a visually appealing and easy to use UI. ● To gracefully handle errors and display error messages to the user.	**Collaborators:** ● interfaceHelper ● handleLoginInterface ● handleLoginLogic ● LoginFrame ● Account ● Main ● loginFormQueries

Class name: LoginFrame	
Parent Class (if any): JFrame **Subclasses (if any):** None	
Responsibilities: ● To be a fitting background frame for the Login Form with correct JFrame configurations.	**Collaborators:** ● LoginForm

Class name: interfaceHelper	
Parent Class (if any): None **Subclasses (if any):** None	
Responsibilities: ● To provide error-free and useful helper methods for the various UI components within the project.	**Collaborators:** ● LoginForm

Class name: passwordHashing	
Parent Class (if any): None **Subclasses (if any):** None	
Responsibilities: ● To provide functionality for hashing passwords and making them secure. ● To provide functionality for comparing passwords against hashed passwords correctly.	**Collaborators:** ● loginFormQueries

Class name: Account	
Parent Class (if any): None **Subclasses (if any):** None	
Responsibilities: ● To provide an organised model of a user Account within the system.	**Collaborators:** ● LoginForm ● CodeEscMenu ● handleLoginLogic

● To be a pathway between the Login Form and the main menu of the program (CodeEscMenu).	

Class name: levelTimeQueries	
Parent Class (if any): None **Subclasses (if any):** None	
Responsibilities: ● To handle database operations related to tracking and retrieving the time a user takes to complete levels within the project. ● To insert a new time entry for a user and level. ● To retrieve the best (minimum) time a user has achieved for a specific level.	**Collaborators:** ● Connection ● PreparedStatement ● ResultSet

Class name: loginFormQueries	
Parent Class (if any): None **Subclasses (if any):** None	
Responsibilities: ● To provide methods that can query the SQL database used for the project for login and sign-up reasons.	**Collaborators:** ● LoginForm ● passwordHashing ● handleLoginLogic

Class name: handleLoginInterface	
Parent Class (if any): None **Subclasses (if any):** None	
Responsibilities: ● To provide methods that alter the LoginForm's UI based on needed functionality.	**Collaborators:** ● LoginForm

Class name: handleLoginLogic	
Parent Class (if any): None **Subclasses (if any):** None	
Responsibilities: ● To provide methods that perform various tasks on the backend to do with the database, and to then return those results to the LoginForm or Account.	**Collaborators:** ● LoginForm ● Account ● loginFormQueries

Class name: Main	
Parent Class (if any): None **Subclasses (if any):** None	
Responsibilities: ● To provide an entry point for the CodeEsc project that correctly launches the program. ● To provide a connection to the database for the rest of the program to use throughout the remainder of the program flow.	**Collaborators:** ● LoginForm

Class name: CodeEscLevelOne	
Parent Class (if any): JFrame **Subclasses (if any):** None	
Responsibilities: ● To display the first level of the game ● To handle level-specific interactions and logic. ● To update the database with user's progress	**Collaborators:** ● CodeEscMenu

Class name: handleMenuInterface
Parent Class (if any): None **Subclasses (if any):** None

Responsibilities: ● To manage the transitions between the main menu and other menu interfaces (e.g. level selection) ● To control visibility of panels in the main menu UI	**Collaborators:** ● CodeEscMenu

Class name: handleLevelInterface	
Parent Class (if any): None **Subclasses (if any):** None	
Responsibilities: ● To modify UI elements for levels ● To overlay the buttons on interactive objects in a level to emulate interactions	**Collaborators:** ● CodeEscLevelOne

Class name: handleLevelOneLogic	
Parent Class (if any): None **Subclasses (if any):** None	
Responsibilities: ● To manage game logic specific to level one, such as puzzles and interactions. ● To handle interactions tied to interactive elements ● To generate random puzzles for level challenges	**Collaborators:** ● CodeEscLevelOne ● puzzleHelper

Class name: puzzleHelper	
Parent Class (if any): None **Subclasses (if any):** None	
Responsibilities: ● To provide utility methods for puzzles ● To scramble characters in a given word to generate a word unscrambling puzzle	**Collaborators:** ● handleLevelOneLogic

Class name: ButtonPanel	
Parent Class (if any): JPanel **Subclasses (if any):** None	
Responsibilities: ● To create a custom panel for the game's main menu with vertically arranged buttons. ● Provides user interface controls for the game, including buttons for "Play", "Settings", and "Leaderboard". ● Manages button actions, such as transitioning to the level selection screen or displaying dialogs for upcoming features.	**Collaborators:** ● handleMenuInterface ● CodeEscMenu ● CodeEscLevels ● JButton, JOptionPane ● Font, Dimension

Class name: CodeEscLevels	
Parent Class (if any): JPanel **Subclasses (if any):** None	
Responsibilities: ● To display the level selection menu for the CodeESC game. ● Provides a "Go Back to Menu" button for returning to the main menu. ● Manages level buttons, including Level 1 and Level 2 (with Level 2 being a feature for a future sprint). ● Configures the layout and appearance of the level panel and buttons.	**Collaborators:** ● handleMenuInterface ● CodeEscMenu ● LevelPanel ● JButton, JOptionPane ● Font, Color

Class name: CodeEscMenu	
Parent Class (if any): JFrame **Subclasses (if any):** None	
Responsibilities: ● To set up the main menu window for the CodeESC game. ● Initializes and configures the GUI components, including the background image, button panel, and help panel.	**Collaborators:** ● Account ● CodeEscLevels ● ButtonPanel ● HelpPanel ● BackgroundPanel

● Manages the transition between the home menu and the level selection menu. ● Holds the user account and manages the play menu flag.	● ImageIcon, Image, JFrame, JPanel

Class name: HelpPanel	
Parent Class (if any): JPanel **Subclasses (if any):** None	
Responsibilities: ● To create a panel containing a "Help" button that displays a help message when clicked. ● Provides a simple user interface for users to get instructions or information about the game. ● Configures the layout and appearance of the help button.	**Collaborators:** ● JButton ● JOptionPane ● Font, Dimension

Class name: LevelPanel	
Parent Class (if any): JButton **Subclasses (if any):** None	
Responsibilities: ● To represent a panel that acts as a button but functions more like a container, displaying detailed information about a specific game level. ● Displays the best time a user has achieved on the level, along with a visually appealing UI using JLabel and background images. ● Updates and displays the best time for the associated level.	**Collaborators:** ● Account ● levelTimeQueries ● BackgroundPanel ● JLabel ● JButton

Class name: levelTimer	
Parent Class (if any): JLabel **Subclasses (if any):** None	
Responsibilities: ● To create and manage a timer that tracks the amount of time a user has spent on a	**Collaborators:** ● Timer ● ActionListener

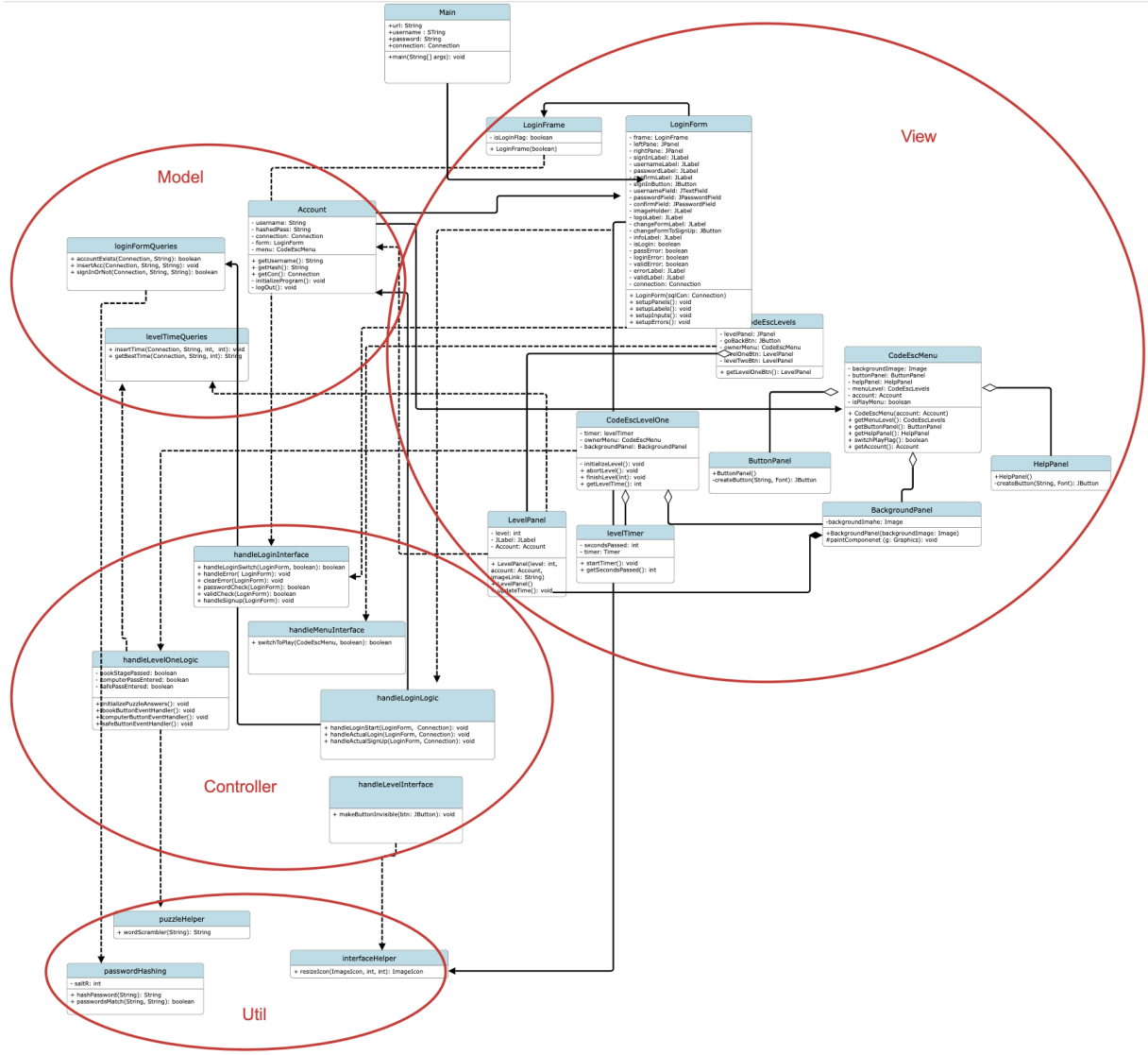
level. ● The timer is displayed as a JLabel, automatically updating to show the time in a MM:SS format. ● Allows other components to start the timer and retrieve the elapsed time in seconds.	● Font, Color

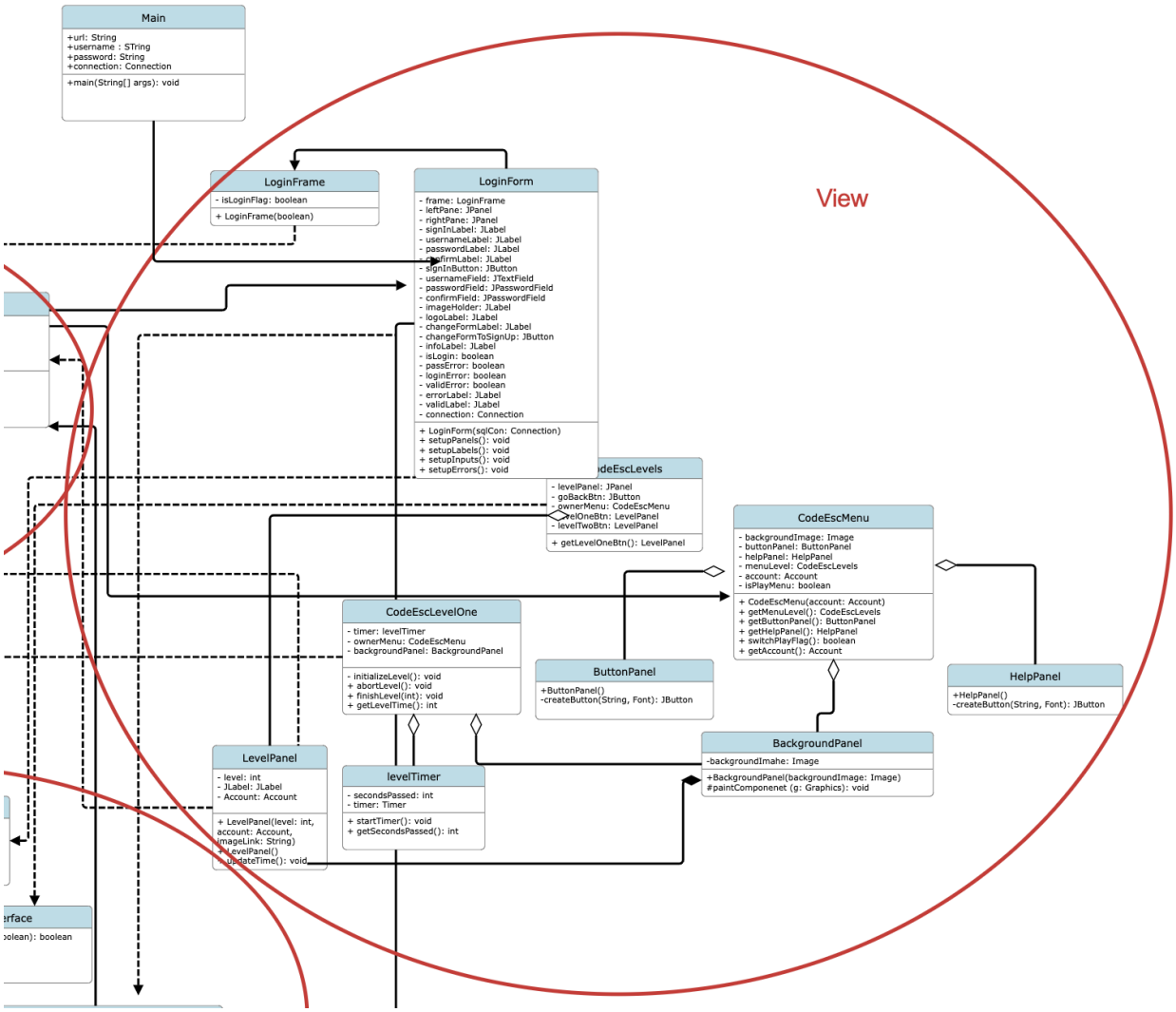
System Interaction with Environment

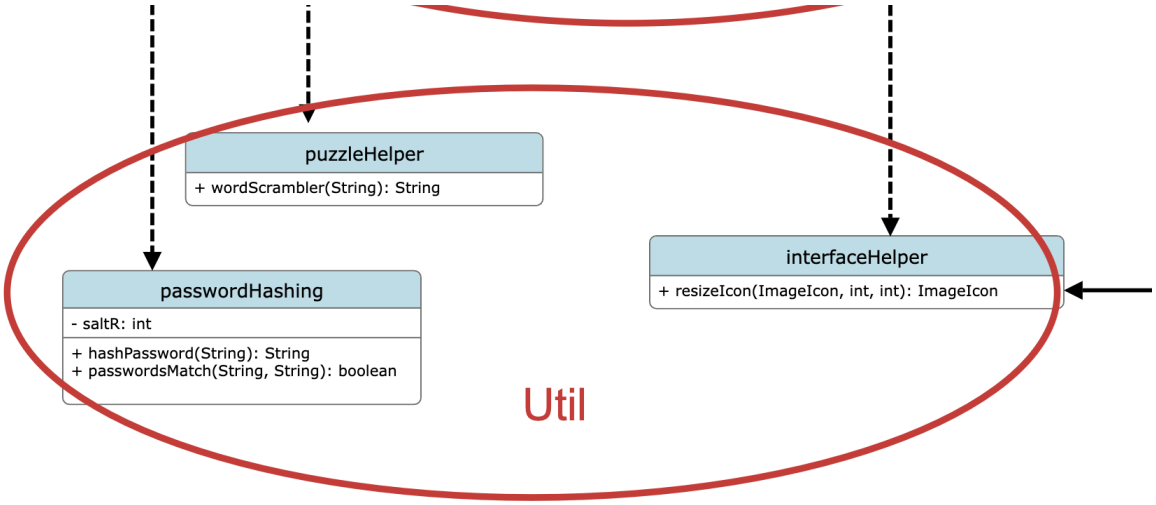
Dependencies and Assumptions

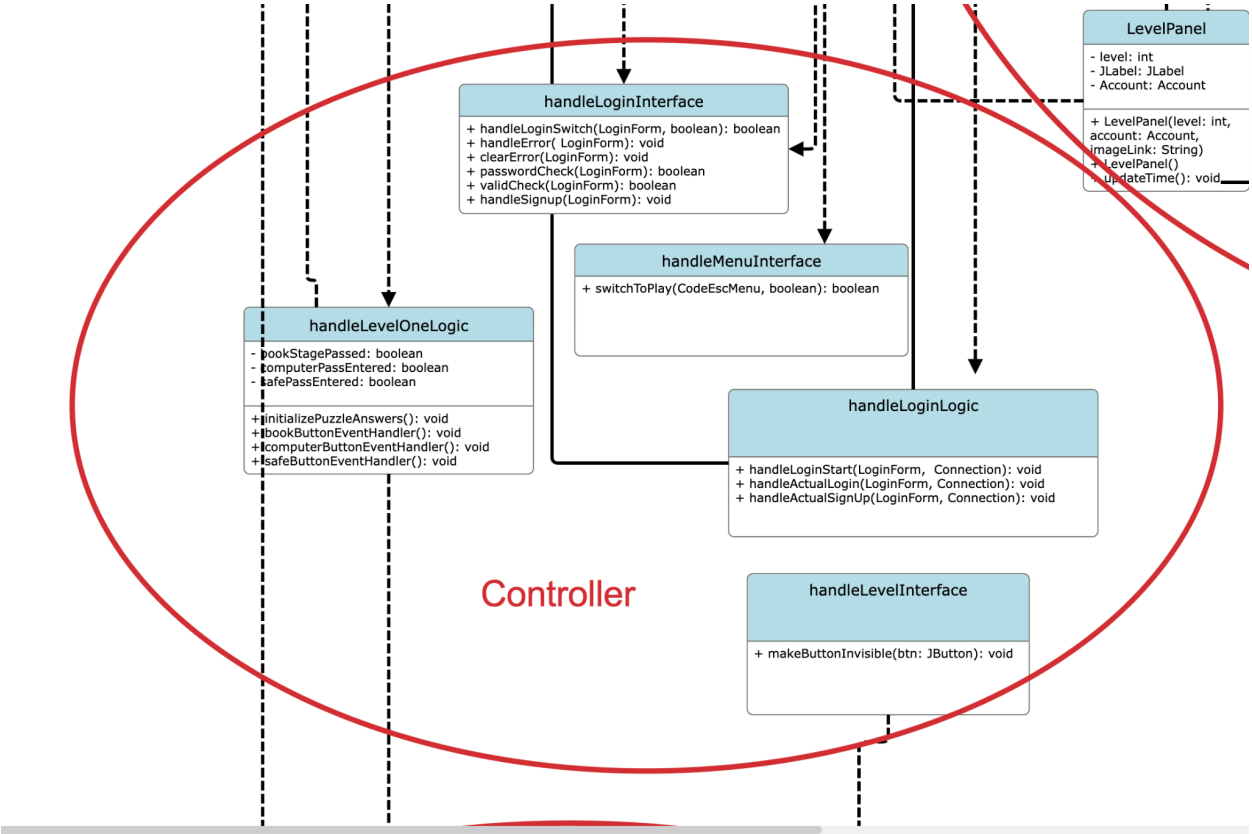
1. **Operating System:**
    - The CodeEsc application is designed to run on multiple operating systems, including Windows, macOS, and Linux.
    - The CodeEsc application is not designed to run on mobile devices such as Android or IOS devices.
2. **Programming Language:**
    - The application is developed using Java, so it is assumed that whatever OS is running this program has either the JRE (Java Runtime Environment) and/or JDK (Java Development Kit) installed to be able to compile and run the application.
3. **Virtual Machine:**
    - Due to the application's nature of being a simple Java program with its used libraries/dependencies included, it can run on any VM that has a supported OS and the right JDK/JRE versions installed.
4. **Database:**
    - This application uses an SQL database for all of its data storage, however the SQL database used for this project is running on a 24/7 server. This means that the user does not have to install or configure anything for the database to work properly, the user just has to have an internet connection while running the application.
5. **Network Configuration:**
    - Since this application uses a database that is hosted on an online server, the user of the application must have a working internet connection. The network configuration is not important as long as they can open commonly used websites on a browser at minimum.
    - An internet connection that is too slow may cause the application to slow down due to a slow connection between the application and the database.

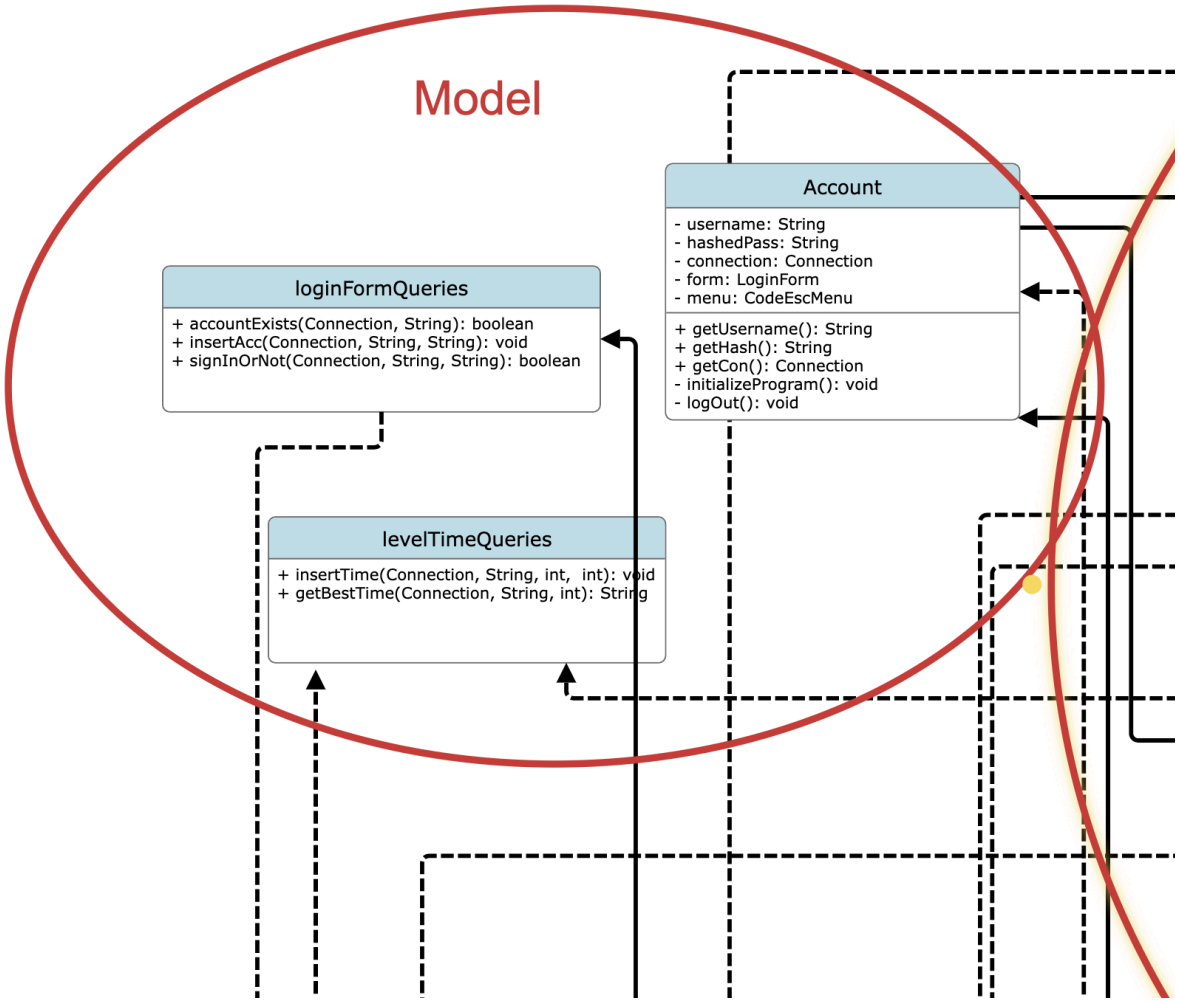
System Architecture




Main
+url: String
+username : STring
+password: String
+connection: Connection
+main(String[] args): void
LoginFrame
- isLoginFlag: boolean
+ LoginFrame(boolean)
LoginForm
- frame: LoginFrame
- leftPane: JPanel
- rightPane: JPanel
- signInLabel: JLabel
- usernameLabel: JLabel
- passwordLabel: JLabel
- confirmLabel: JLabel
- signInButton: JButton
- usernameField: JTextField
- passwordField: JPasswordField
- confirmField: JPasswordField
- imageHolder: JLabel
- logoLabel: JLabel
- changeFormLabel: JLabel
- changeFormToSignUp: JButton
- infoLabel: JLabel
- isLogin: boolean
- passError: boolean
- loginError: boolean
- validError: boolean
- errorLabel: JLabel
- validLabel: JLabel
- connection: Connection
+ LoginForm(sqlCon: Connection)
+ setupPanels(): void
+ setupLabels(): void
+ setupInputs(): void
+ setupErrors(): void
View
CodeEscLevels
- levelPanel: JPanel
- goBackBtn: JButton
- ownerMenu: CodeEscMenu
- levelOneBtn: LevelPanel
- levelTwoBtn: LevelPanel
+ getLevelOneBtn(): LevelPanel
CodeEscMenu
- backgroundImage: Image
- buttonPanel: ButtonPanel
- helpPanel: HelpPanel
- menuLevel: CodeEscLevels
- account: Account
- isPlayMenu: boolean
+ CodeEscMenu(account: Account)
+ getMenuLevel(): CodeEscLevels
+ getButtonPanel(): ButtonPanel
+ getHelpPanel(): HelpPanel
+ switchPlayFlag(): boolean
+ getAccount(): Account
CodeEscLevelOne
- timer: levelTimer
- ownerMenu: CodeEscMenu
- backgroundPanel: BackgroundPanel
- initializeLevel(): void
+ abortLevel(): void
+ finishLevel(int): void
+ getLevelTime(): int
ButtonPanel
+ButtonPanel()
-createButton(String, Font): JButton
HelpPanel
+HelpPanel()
-createButton(String, Font): JButton
BackgroundPanel
-backgroundImahe: Image
+BackgroundPanel(backgroundImage: Image)
#paintComponenet (g: Graphics): void
LevelPanel
- level: int
- JLabel: JLabel
- Account: Account
+ LevelPanel(level: int, account: Account, imageLink: String)
+ LevelPanel()
- updateTime(): void
levelTimer
- secondsPassed: int
- timer: Timer
+ startTimer(): void
+ getSecondsPassed(): int


puzzleHelper
+ wordScrambler(String): String
passwordHashing
- saltR: int
+ hashPassword(String): String
+ passwordsMatch(String, String): boolean
interfaceHelper
+ resizeIcon(ImageIcon, int, int): ImageIcon
Util


LevelPanel
- level: int
- JLabel: JLabel
- Account: Account
+ LevelPanel(level: int, account: Account, imageLink: String)
+ LevelPanel()
+ updateTime(): void
handleLoginInterface
+ handleLoginSwitch(LoginForm, boolean): boolean
+ handleError(LoginForm): void
+ clearError(LoginForm): void
+ passwordCheck(LoginForm): boolean
+ validCheck(LoginForm): boolean
+ handleSignup(LoginForm): void
handleMenuInterface
+ switchToPlay(CodeEscMenu, boolean): boolean
handleLevelOneLogic
- bookStagePassed: boolean
- computerPassEntered: boolean
- safePassEntered: boolean
+initializePuzzleAnswers(): void
+bookButtonEventHandler(): void
+computerButtonEventHandler(): void
+safeButtonEventHandler(): void
handleLoginLogic
+ handleLoginStart(LoginForm, Connection): void
+ handleActualLogin(LoginForm, Connection): void
+ handleActualSignUp(LoginForm, Connection): void
Controller
handleLevelInterface
+ makeButtonInvisible(btn: JButton): void


Model
Account
- username: String
- hashedPass: String
- connection: Connection
- form: LoginForm
- menu: CodeEscMenu
+ getUsername(): String
+ getHash(): String
+ getCon(): Connection
- initializeProgram(): void
- logOut(): void
loginFormQueries
+ accountExists(Connection, String): boolean
+ insertAcc(Connection, String, String): void
+ signInOrNot(Connection, String, String): boolean
levelTimeQueries
+ insertTime(Connection, String, int, int): void
+ getBestTime(Connection, String, int): Strin

System Decomposition

Component Roles in High-Level Architecture

1. **Main Component**
    - *Role:* Entry point and initializer.
    - *Detailed Design:*
        - Database Initialization: The Main class establishes a database connection, which is passed to other components that interact with data. It provides essential database parameters (e.g., URL, username, password) securely.
        - Interface Launching: Launches LoginForm, setting the starting point for user interaction.
2. **Model Components**
    - *Role:* Data management and business logic.
    - *Detailed Design:*
        - Account: Manages account data, including user attributes and session information. Acts as a central repository for user-specific operations.
        - loginFormQueries: Executes database queries for account validation, lookup, and insertion. It relies on secure methods provided by passwordHashing to verify user credentials.
        - levelTimeQueries: Manages database queries for storing and retrieving level completion times.
3. **Controller Components**
    - *Role:* Coordinate between Model and View, manage logic and flow.
    - *Detailed Design:*
        - handleLoginLogic: Handles business logic for login processes, such as validating user credentials, creating sessions, and managing login state.
        - handleLoginInterface: Manages the display of views, updating the interface based on user actions or model responses.
        - handleMenuInterface: Handles transitions within the main menu, including switching between levels and help panels.
        - handleLevelOneLogic: Implements Level One's core gameplay logic, including puzzles like guessing numbers, entering passwords, and unscrambling words.
        - handleLevelInterface: Provides utility methods for interactive UI elements in escape room levels, such as making buttons visually invisible for object overlays.
4. **View Components**
    - *Role:* Present the user interface, display data, and gather user input.
    - *Detailed Design:*

        - LoginForm: The initial user interface for logging in or signing up. It communicates user inputs and displays errors or success messages from the Controller.
        - CodeEscMenu: Provides a secondary interface post-login, containing panels like BackgroundPanel, ButtonPanel, and HelpPanel, each managing distinct UI components.
        - LoginFrame: A subclass of JFrame that sets up the frame for login and sign-up pages, configuring dimensions, titles, and close operations.
        - CodeEscLevels: Provides the interface for selecting levels, including buttons to start levels or return to the main menu.
        - CodeEscLevelOne: Implements the Level One UI, featuring interactive components like buttons and puzzles. Displays a timer and uses handleLevelOneLogic for gameplay logic.
        - levelTimer:A timer component that displays elapsed time in levels, helping users track their performance.

5. **Utility Components**
    - *Role:* provide essential supporting functionalities to enhance the overall efficiency, security, and usability of the system.
    - *Detailed Design:*
        - passwordHashing: Used by loginFormQueries for secure password handling, this utility hashes passwords before they are stored or compared with the database.
        - interfaceHelper: Assists the View components, providing image resizing and formatting to ensure a polished user experience.
        - puzzleHelper: Provides methods for puzzles, such as scrambling words for word-based challenges in Level One.

Error Handling Strategy

In any application errors can happen, and they can happen for various reasons. This is something we take into consideration as we develop our application for CodeEsc, so we have developed various ways to handle these errors.

To start off, let's talk about non user-interaction errors. The application needs access to a database server to actually launch itself, so if the user does not have an internet connection or the connection just doesn't work for whatever reason then the application will not launch and an error message will be printed to the console about the connection failing which is easy to understand. If the database connection ever stops working while the application is running, then

error messages will be printed to the console for the user to see while the application still retains some functionality and doesn't completely fail.

To continue, let's talk about user-interaction errors, these are ones that include things like user input for their account information on a sign-up or login page, which are two big features added in Sprint 1. We have considered these errors as well, and have added validation for user input to match certain length restrictions that are displayed as error messages in the UI if not followed, we have also added a confirm password feature that always checks that a user has entered two matching passwords whether it's a sign-up or login event they are trying to push through. There are many errors considered in the login/sign-up process when dealing with the database as well, such as wrong username or password, or an account existing with the specified username already, and those all display their own unique error messages to make an easy to follow process for the user.

When we develop this application, we make sure that common and uncommon errors are handled in a way that keeps the user experience as pleasant as possible, and our above considerations show this. For our future sprints we plan to add more validation to user input for the sign-up/login process, and to show more clear error messages for database connections that use pop ups instead of console messages to make it more easily understandable for the user.