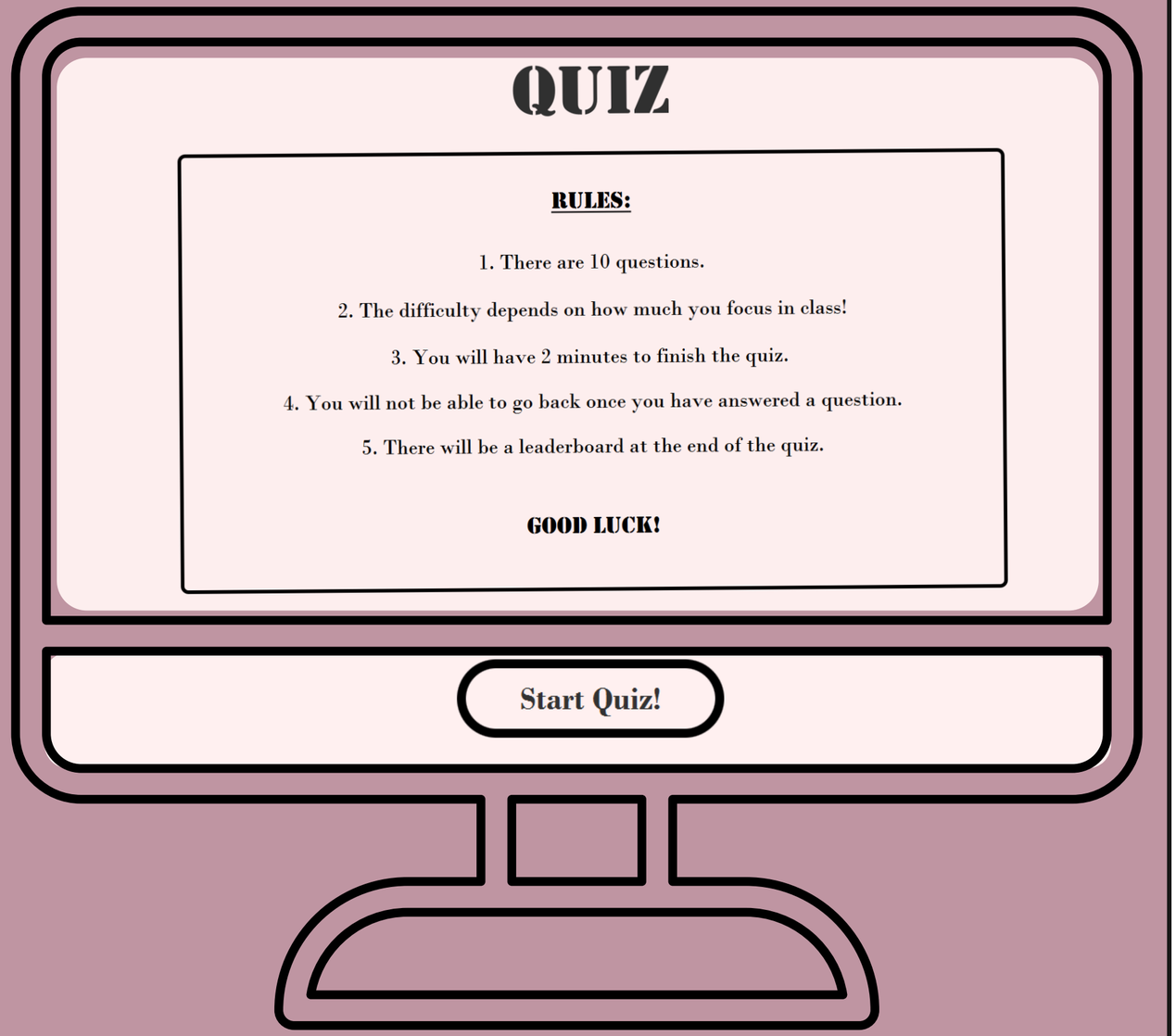
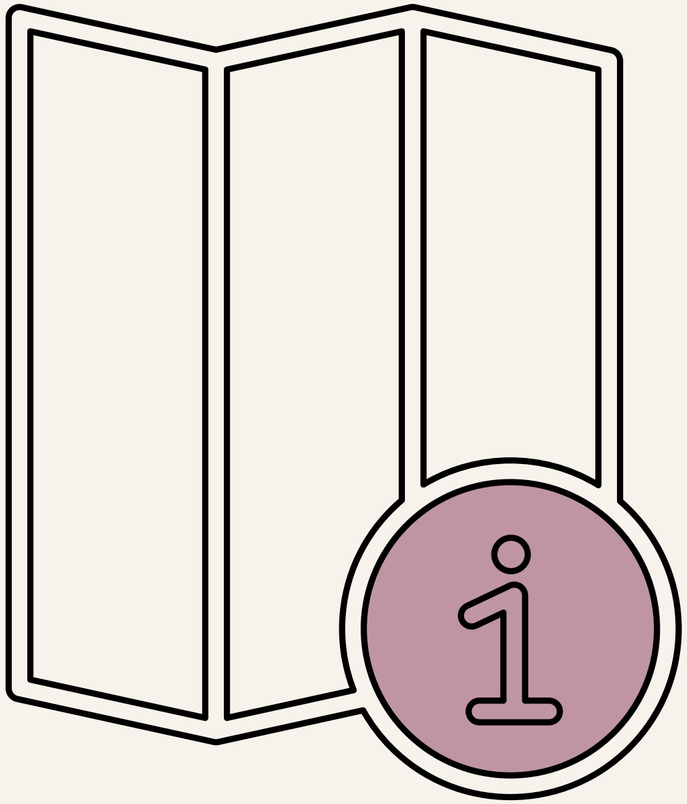
quiz gui

research & desgin

Haneen 418006336
Heer 752000815
Nastasya 764000305



Our Goal



A fun, simple and non-depressing quiz interface.

We wanted a quiz that looks good and feels good to use!

Best UI / UX practices

Accesibility

In the aspect of quizzes, the user already has to pour all of their attention to the questions, why make the user's life even harder if the feature of accessibility doesnt exist?

Simplicity

Users should focus on the content without distractions, and also without getting overwhelmed with the amount of things going on.

Visual Hierarchy

Notice how the 3 big words actually got your attention? That's the thing with visual hierarchy, you need to make that the most important elements are easy to find!

Competitive Research

Kahoot very fun and well structured but complex

Quizzez good feedback system, but too many options
<might overwhelm>

Blooket interactive and game-like, but too advanced for a simple quiz

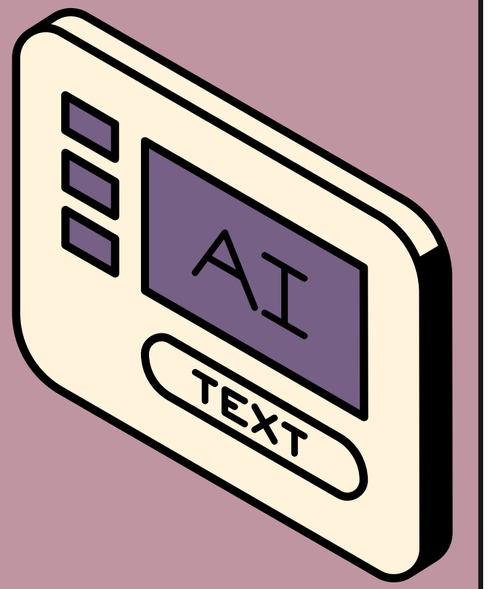
Challenges and issues we faced

1. Alignment & Layout Issues: Elements didn’t align properly

2. Interactivity vs. Simplicity: Balancing engagement with usability

3. Radio Button Bug: Multiple selections were possible (Fixed with ToggleGroup)

4. Scene Switching: Managing transitions between screens



QUIZ

RULES:

1. There are 10 questions.
2. The difficulty depends on how much you focus in class!
3. You will have 2 minutes to finish the quiz.
4. You will not be able to go back once you have answered a question.
5. There will be a leaderboard at the end of the quiz.

GOOD LUCK!

USERNAME:

Start Quiz!

Quiz Completed!

View Leaderboard

2. Which keyword is used to create a subclass?

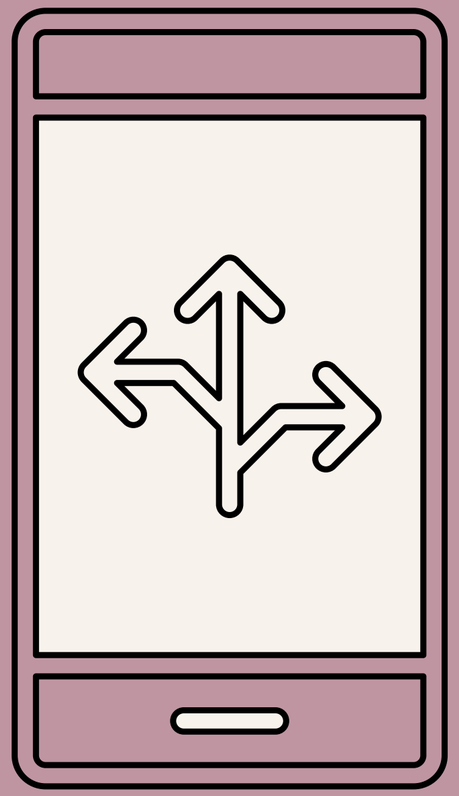
- Extends
- Implements
- Inherits
- Sub

Next

simple yet efficient

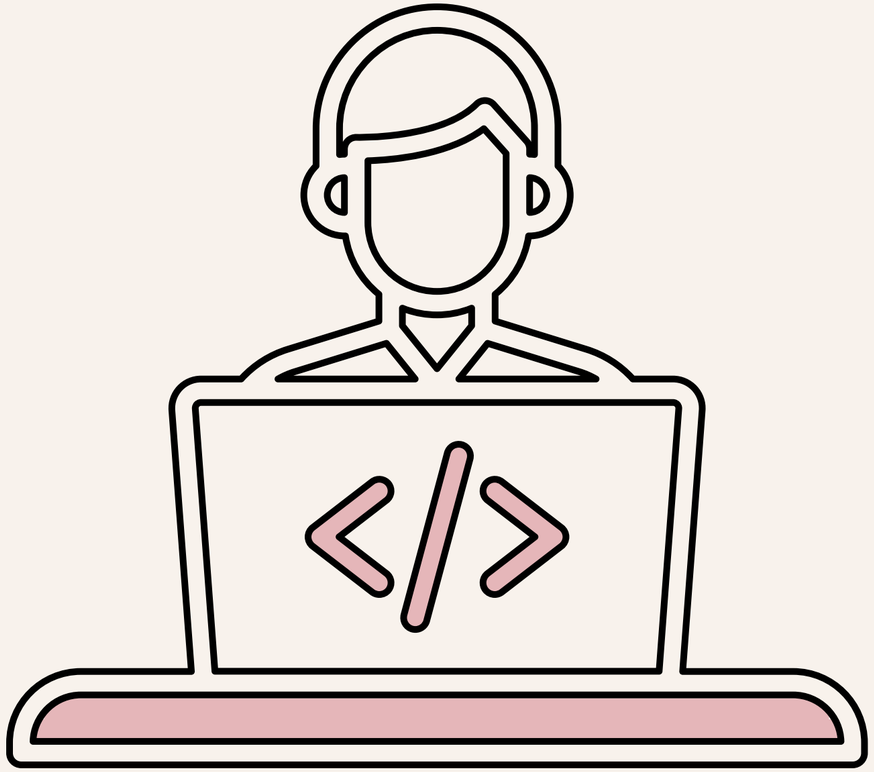
what makes our GUI unique

1. its designed to be stress free
2. its aesthetic but not distracting
3. Simple, no clutter



our GUI quiz isn’t just another quiz, its a better experience:)

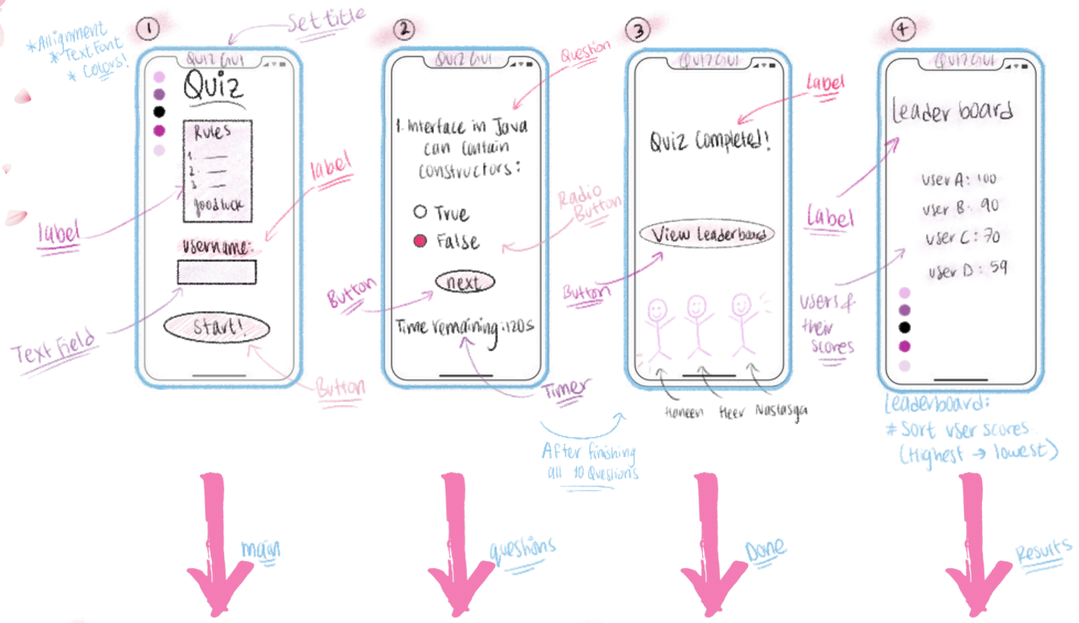
future improvements

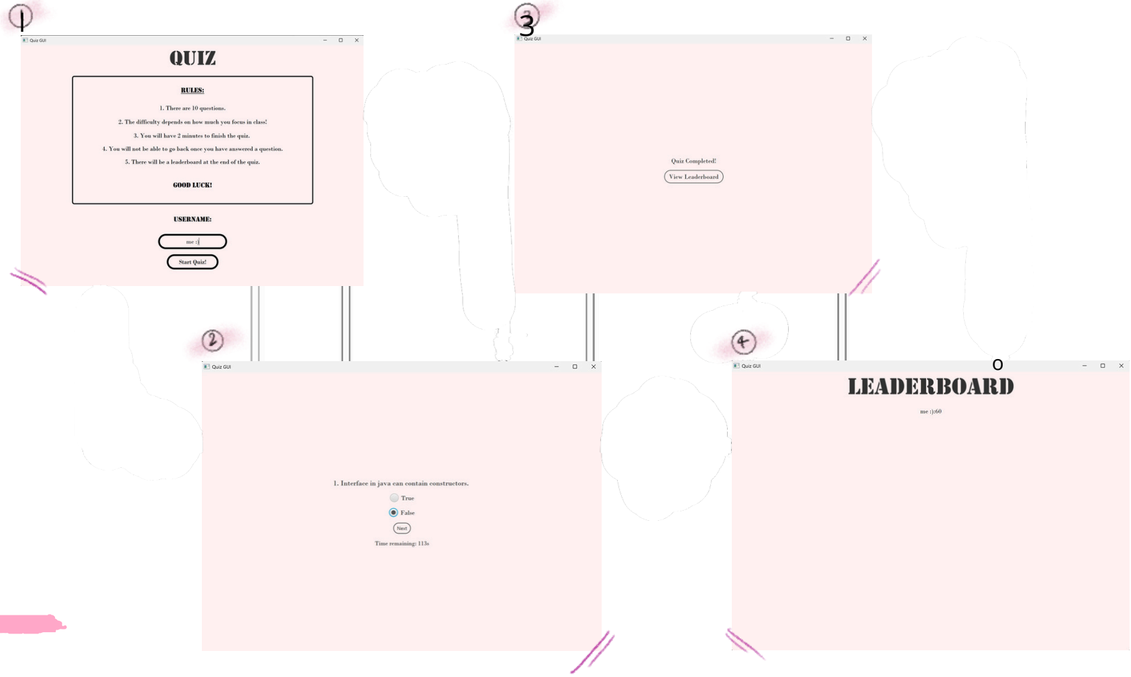


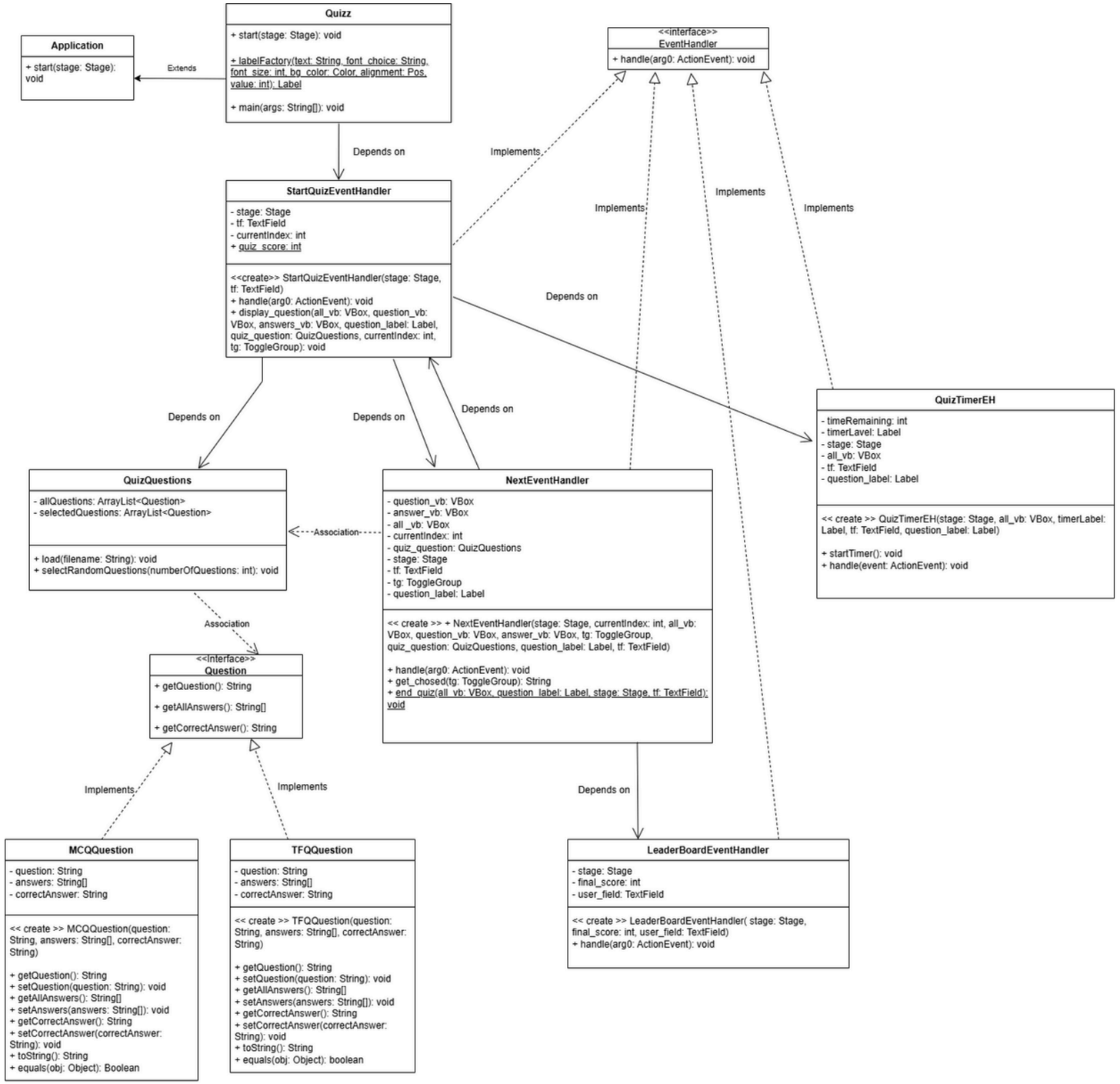
Maybe adding something like feedbacks after the quiz, or even making it more interactive while its still simple

GUI Sketch

{ group 1 }






Quizz
+ start(stage: Stage): void
+ labelFactory(text: String, font_choice: String, font_size: int, bg_color: Color, alignment: Pos, value: int): Label
+ main(args: String[]): void
Application
+ start(stage: Stage): void
Extends
<<interface>>
EventHandler
+ handle(arg0: ActionEvent): void
Depends on
Implements
Implements
Implements
Implements
StartQuizEventHandler
- stage: Stage
- tf: TextField
- currentIndex: int
+ quiz_score: int
<<create>> StartQuizEventHandler(stage: Stage, tf: TextField)
+ handle(arg0: ActionEvent): void
+ display_question(all_vb: VBox, question_vb: VBox, answers_vb: VBox, question_label: Label, quiz_question: QuizQuestions, currentIndex: int, tg: ToggleGroup): void
Depends on
Depends on
Depends on
Depends on
QuizTimerEH
- timeRemaining: int
- timerLavel: Label
- stage: Stage
- all_vb: VBox
- tf: TextField
- question_label: Label
<< create >> QuizTimerEH(stage: Stage, all_vb: VBox, timerLabel: Label, tf: TextField, question_label: Label)
+ startTimer(): void
+ handle(event: ActionEvent): void
QuizQuestions
- allQuestions: ArrayList<Question>
- selectedQuestions: ArrayList<Question>
+ load(filename: String): void
+ selectRandomQuestions(numberOfQuestions: int): void
Association
NextEventHandler
- question_vb: VBox
- answer_vb: VBox
- all _vb: VBox
- currentIndex: int
- quiz_question: QuizQuestions
- stage: Stage
- tf: TextField
- tg: ToggleGroup
- question_label: Label
<< create >> + NextEventHandler(stage: Stage, currentIndex: int, all_vb: VBox, question_vb: VBox, answer_vb: VBox, tg: ToggleGroup, quiz_question: QuizQuestions, question_label: Label, tf: TextField)
+ handle(arg0: ActionEvent): void
+ get_chosed(tg: ToggleGroup): String
+ end_quiz(all_vb: VBox, question_label: Label, stage: Stage, tf: TextField): void
Association
<<Interface>>
Question
+ getQuestion(): String
+ getAllAnswers(): String[]
+ getCorrectAnswer(): String
Implements
Implements
Depends on
MCQQuestion
- question: String
- answers: String[]
- correctAnswer: String
<< create >> MCQQuestion(question: String, answers: String[], correctAnswer: String)
+ getQuestion(): String
+ setQuestion(question: String): void
+ getAllAnswers(): String[]
+ setAnswers(answers: String[]): void
+ getCorrectAnswer(): String
+ setCorrectAnswer(correctAnswer: String): void
+ toString(): String
+ equals(obj: Object): Boolean
TFQQuestion
- question: String
- answers: String[]
- correctAnswer: String
<< create >> TFQQuestion(question: String, answers: String[], correctAnswer: String)
+ getQuestion(): String
+ setQuestion(question: String): void
+ getAllAnswers(): String[]
+ setAnswers(answers: String[]): void
+ getCorrectAnswer(): String
+ setCorrectAnswer(correctAnswer: String): void
+ toString(): String
+ equals(obj: Object): boolean
LeaderBoardEventHandler
- stage: Stage
- final_score: int
- user_field: TextField
<< create >> LeaderBoardEventHandler(stage: Stage, final_score: int, user_field: TextField)
+ handle(arg0: ActionEvent): void

Thanks Y'all

Part 2 of our journey

Finalizing CodeBean

After research and prototyping in Assignments 1 and 2, we used feedback, testing, and branding to bring CodeBean to life. Here's how our quiz app evolved from a concept to a fully functional Java application.

New features implemented

What We Added

⏱️ Timer now uses Java threads and auto-submits the quiz

🧪 Added JUnit testing to verify scoring logic and edge cases

🧮 Leaderboard now sorts by score (high to low) and time taken (low to high)

☕ Introduced a coffee bean cursor for a personal GUI touch

Survey Results Summary

Total Responses: 12

Highlights:

✅ All users said the interface was easy to navigate

✅ Questions were clear and understandable

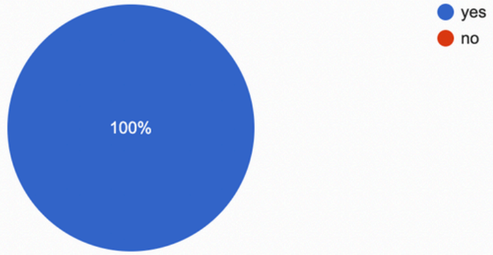
✨ Fix display clarity (questions + answers)

🎨 Multiple people suggested more creativity in design and color

💡 Some wanted to see which answers were wrong

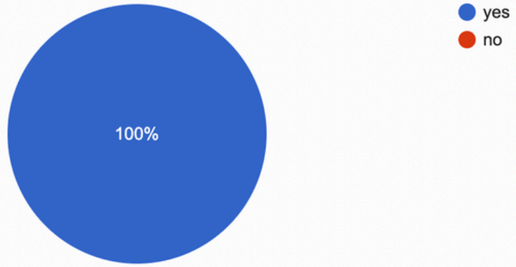
1. Was the quiz interface easy to navigate?

12 responses



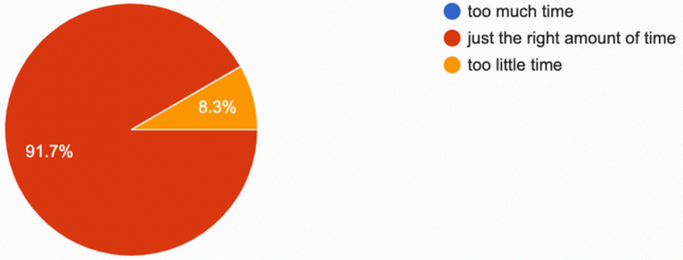
2. Did you find the quiz questions clear and understandable?

12 responses



3. Did the timer (2 minutes) feel appropriate for answering the questions?

12 responses



4. Were there any questions that seemed unclear or unfair? If so, please mention them.

9 responses

- Yea it was fair
- all was clear
- Nah
- Nope, everything was clear
- no questions difficulty was fine
- no
- Jo
- the questions seemed fine
- nope

5. How would you rate the quiz's design and visual appeal?

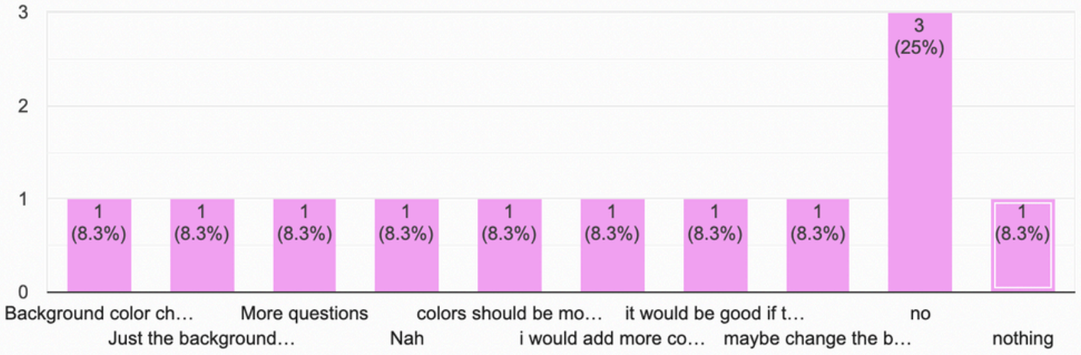
Copy chart

12 responses

6. Do you have any suggestions to improve the quiz?

Copy chart

12 responses



Changes Made Based on Feedback

Improvements After Survey

🎨Changed background color to better reflect user feedback and creativity

❌ Displayed incorrect answers after quiz for user clarity

🕒 Fixed timer to work more smoothly and submit correctly

🖥️ Resized the GUI screen to improve layout and readability

🖱️ Added custom coffee bean cursor to match Java branding

Brand Identity – CodeBean

Introducing: CodeBean ☕🐻

Name: CodeBean

Motto: "Coding never looked this cozy."

CodeBean is a Java-based quiz app built for comfort, clarity, and creativity. Inspired by coffee beans, coding, and Mr. Bean's iconic teddy, it blends logic and lightheartedness to make programming a warmer experience.

Design Theme: Calm, Coffee, Code

Soft warm neutral

Cozy coffee bean color

Light and calming contrast

Mr. Bean inspired pop color

Deep accent tone

Clean, readable text

Quiz GUI

QUIZ

RULES:

1. There are 10 questions.
2. The difficulty depends on how much you focus in class!
3. You will have 2 minutes to finish the quiz.
4. You will not be able to go back once you have answered a question.
5. There will be a leaderboard at the end of the quiz.

GOOD LUCK!

USERNAME:

MrBean

Start Quiz!

NOTE: A 2-minute timer will begin once you press start quiz.

Quiz GUI

FEEDBACK

MrBean, you got 3 question(s) wrong.

1. Which keyword is used to create a subclass?
2. In java, an ArrayList can store primitive data type like int and double directly.
3. What is the output of a System.out.println(10/3); in java?

Quiz GUI

QUESTION 1 OF 10

1. Which keyword is used to create a subclass?

- Extends
- Implements
- Inherits
- Sub

Next

Time left: 110 seconds

Quiz GUI

LEADERBOARD

HeySweetie : 100 points in 11s

THISWORKS : 100 points in 13s

please : 100 points in 13s

View Wrong Questions

Projected Budget

Development Budget

Item	Estimated cost (USD)
UI/UX Design	$300
Java Development	$500
Testing & Debugging	$200
Branding & Logo	$100
Survey Collection	$50
Total	1,150 USD

Difficulties Faced

1. Balancing simplicity and interactivity in GUI design
2. Handling Java thread synchronization for timer functionality
3. Making the leaderboard sort correctly by both score & time
4. Finding a unique and consistent branding identity
5. Coordinating feedback from different users and applying it meaningfully

Future Improvement Ideas

1. Add in-quiz feedback or hints for learning support
2. Track user progress or stats across multiple quiz sessions
3. Implement theme switcher (light/dark/bean mode) for accessibility
4. Make it mobile-friendly or responsive layout
5. Add user login and personalized leaderboards

Reflection & Takeaways

What We Learned

1. Learned how to implement Java threads, sorting, and testing
2. Understood the value of user feedback
3. Found a balance between aesthetics and performance
4. Improved teamwork through real coding collaboration
5. Felt more confident in designing and delivering a full Java application

Evaluating Our Project & Future Goals

How CodeBean Stands Out

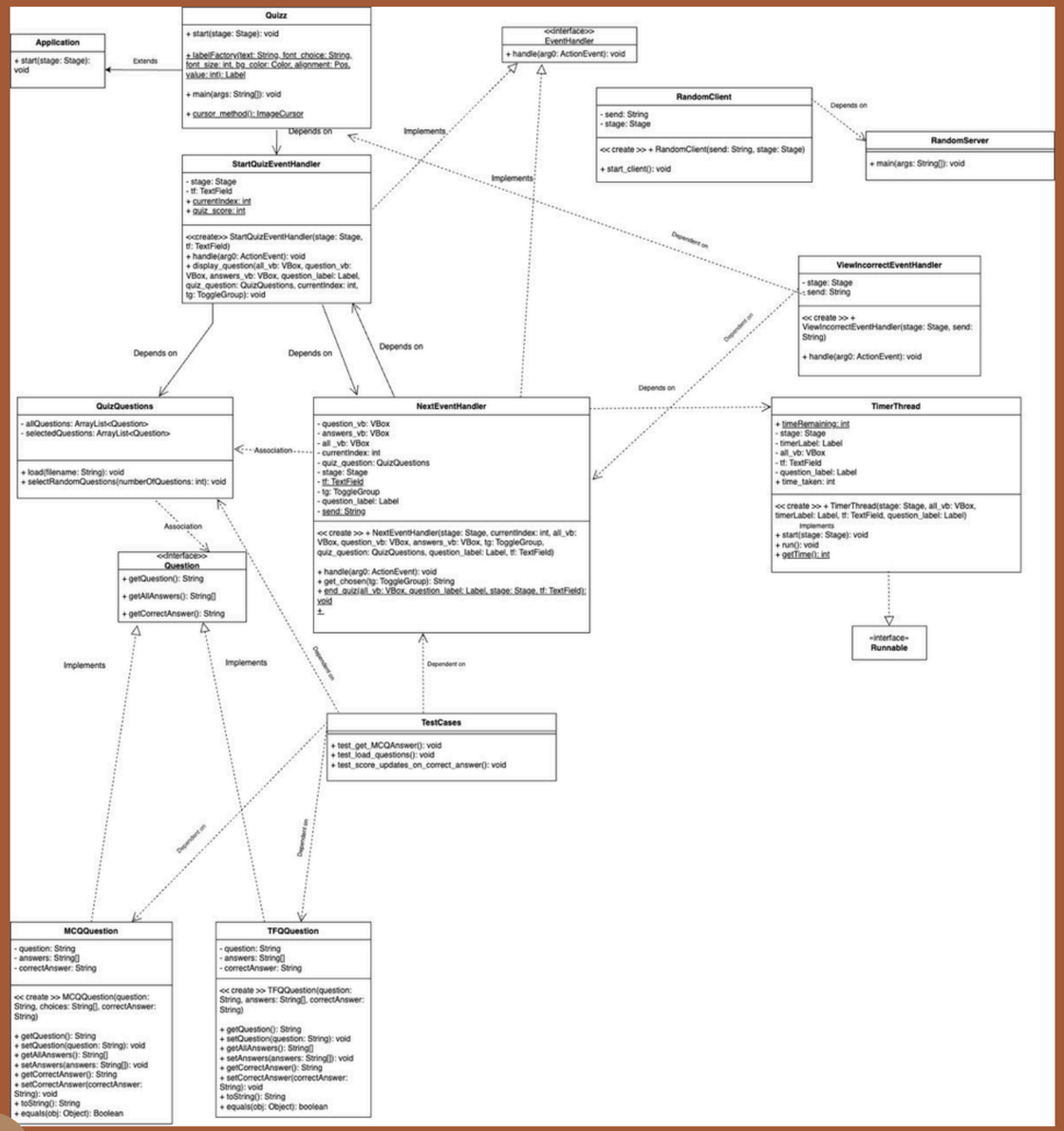
1. Fully themed GUI (cursor, logo, colors)
2. Unique mascot and brand personality
3. Functional leaderboard and timer implementation
4. Clear user feedback integration

What We'd Do Differently

1. Start branding earlier so it reflects deeper in the code
2. Conduct a longer testing period with more users
3. Improve code structure for scalability

Honestly? This project was a struggle. There were times we wanted to scream, quit, or pretend we never signed up for this course. It was stressful, chaotic, and felt like a never-ending checklist. But somehow... we pulled through. We treated this project like our child — annoying, time-consuming, but something we genuinely cared about. And now, seeing how far we've come, we're actually proud. We never thought we'd finish it the way we did — and the journey, as painful as it was, taught us way more than just Java.

Thanks Y'all



New and upgraded uml!