[CreditPath]

[Group 13]

[Alberto Campuzano, Miguel Madrigal, Jaime Morales, Sai Samith Sudini]

Implementation Internals and Architecture Discussion

Our CreditPath educational application was designed following a **component-based architecture** using **React.js**, with strong influences from the **Model-View-Controller (MVC)** pattern discussed in UI Software Architecture lectures.

At a high level:

- **Model:** Represented by the lesson topics and videos, selected dynamically through URL parameters.

- **View:** Encapsulated in React components (`CreditEducationHub`, `StartLesson`), responsible for rendering and user interaction.

- **Controller:** Navigation and user input handling, achieved through modular functions like `handleNavigate()`.

This structure ensures a **clear separation of concerns**, which was emphasized in lecture concepts on interface architecture.

Design Patterns Applied

- **Model-View-Controller (MVC):**
  As taught, MVC organizes applications into models (data), views (presentation), and controllers (logic).
  Our app cleanly separates:

  - Topic data (model),

  - Displaying the lessons (view),

  - Managing user navigation between lessons (controller).

- **Component-Based Hierarchies:**
  Following the idea of **view hierarchies** from lecture slides, we organized our UI into reusable components, with layout and spatial organization controlled using flexbox CSS.
  Each "page" is a container holding buttons, videos, and text components.

- **State and Event Management:**
  In line with **states and events** concepts, our application listens to user actions (clicks) and triggers state changes (navigating between lessons, answering questions).
  We also use event handlers (like `onClick`) to directly tie user events to UI updates.

- **Event Propagation:**
  Events such as button clicks propagate from individual buttons (child components) to navigation functions that affect page-wide (parent) components — matching the **event propagation** model taught in class.

Key Design Decisions

- **Dynamic Content Loading:**
  Instead of separate hard coded pages for each credit lesson, we use URL query parameters (`?topic=APR`) to dynamically update the same `StartLesson` page.

  - This decision aligns with the modular, scalable architectures discussed in UI Software Architecture.

- **Front-End Only Application:**
  We chose a **stateless front-end architecture**, with no back-end, matching early UI prototyping goals.
  All lesson navigation and display logic happen client-side.

- **Minimized Redundancy:**
  Components and CSS styles were reused across multiple pages to maintain a uniform visual hierarchy, in accordance with the **DRY (Don't Repeat Yourself)** principle.

Implementation Challenges and Usability Impacts

During implementation, we faced several issues that impacted usability:

- **Topic Loss During Navigation:**
  Early versions lost user-selected topic data when navigating.
  - Solution: Incorporated URL query strings and parsed them dynamically on page load.

- **Generic Button Labels:**
  Initial answer buttons labeled "Answer 1" and "Answer 2" confused users.
  - Planned improvement: Use meaningful labels that reflect question content like simple true and false.

- **Navigation Visibility:**
  Some users overlooked the back button due to visual similarity with other buttons.

- Solution: Improve button visibility and affordances through differentiated styling.

These issues highlight real-world challenges in **state management**, **event handling**, and **visual affordance design** — major topics from UI Software Architecture lectures.

Connection to UI Software Architecture Concepts

Concept	How It Applies
Model-View-Controller (MVC)	Clear separation between topic data, UI presentation, and user input handling
Component Hierarchies	Each page structured into small, composable React components
State and Events	State changes based on user input (button clicks trigger topic loading)
Event Propagation	Clicks and navigation events propagate cleanly through the component tree
View Layout and Organization	CSS flexbox ensures spatial organization of components, matching layout rules

We also observed that **strict MVC** can be challenging for highly interactive web apps, as discussed in the lecturer's disadvantages of MVC — leading us to adopt a **light MVVM style** where the view and logic are often tied inside React components.

Our CreditPath app embraces modern UI architectural principles: modular design, separation of concerns, event-driven updates, and component reusability. Challenges encountered were addressed iteratively, improving both system organization and user experience — a practical reflection of core UI Software Architecture concepts.

User Testing Report

Introduction

We conducted a user testing session for **CreditPath**, an educational application designed to help young adults better understand credit concepts and credit card management.
The purpose of this user test was to evaluate the usability of the prototype through a **think-aloud user testing method**, where participants verbalized their thoughts while completing tasks.
We tested our final implementation prototype with **five participants**, and collected qualitative and quantitative feedback based on task completion, time taken, and errors encountered.

Tasks and Success Criteria

Participants were asked to complete a series of tasks that reflect common use cases within the CreditPath application.
We defined success for each task based on whether the user could complete the action correctly without assistance and within a reasonable time frame.

The tasks were:

1. **Select a lesson topic from the Hub:**
   Users needed to choose one of the available topics (such as APR, Utilization, Interest Rates, or Annual Fees) and navigate to the corresponding lesson page.
   Success Criteria: The selected topic should match the lesson page title.

2. **Watch a lesson video and select an answer:**
   After reaching the lesson page, users were instructed to watch a short portion of the educational video and then click one of the provided answer buttons.
   Success Criteria: The user should watch part of the video and select one of the two answer options without external guidance.

3. **Return to the Hub using the navigation buttons:**
   After interacting with a lesson, participants needed to return to the Education Hub by clicking the "Back" button.
   Success Criteria: The user must correctly navigate back to the Hub page.

4. **Start the quiz:**
   Finally, users were asked to click the "Take Quiz" button from the Hub and reach the quiz introduction page.

Success Criteria: The user should successfully navigate to the quiz section.

Testing Methodology

We utilized the **think-aloud protocol** during testing. Participants were asked to verbalize their thoughts, reactions, and decisions as they interacted with the app.
If a participant was silent for more than 10 seconds, we reminded them to "Please remember to think aloud."
If they appeared stuck for more than 30 seconds, we asked, "Can you tell me what you are trying to do?"
After 90 seconds of no progress, we gently ended the current task and moved on to the next one.
Each participant completed all tasks sequentially, followed by a short feedback questionnaire.

Number of Participants

A total of **five users** participated in the usability study.
Participants were undergraduate students aged 19–24 years, selected for their familiarity with using basic websites but without advanced knowledge of credit management.

User ID	**Task ID**	**Success**	**Failure**	**Amount of time (in seconds)**	**Behaviors**	**Intention**	**Error**
User 1	1	X		45 seconds	Smooth Selection	Pick a lesson	None
User 1	2	X		1 minute	Clicked answer correctly	Complete Lesson	None
User 2	1	X		30 seconds	Missed Back button	Return to homepage	Didn't see back button
User 2	2		X	2 minutes	Didn't find answer immediately	Complete lesson	Answer buttons unclear
User 3	1	X		30 sec	Smooth	Pick a lesson	None
User 3	2	X		1 minute	Smooth	Complete lesson	None
User	1	X		40 sec	Confident	Pick a lesson	None

4							
User 4	2		X	3 minute	Hesitated, hovered	Complete lesson	Buttons confusing
User 5	1	X		30 sec	Smooth	Pick a lesson	None
User 5	2	X		2 min	Smooth	Complete lesson	None

Data Logging Summary

During the user testing session, we logged participant performance for two main tasks: selecting a lesson topic and completing a lesson by watching a video and answering a question.

For **Task 1 (Select Lesson)**:

- All five users successfully selected a lesson from the main hub.
- Completion times ranged from **30 seconds to 45 seconds**.
- Users reported smooth and confident experiences during this step.
- No major errors were observed for Task 1.

For **Task 2 (Watch Video and Answer)**:

- 4 out of 5 users successfully completed the task by watching the lesson video and selecting an answer.
- Completion times varied between **1 to 3 minutes**.
- Two users initially hesitated:
  - One user took **2 minutes** and struggled to find the answer buttons because the labels were unclear.
  - Another user hesitated for **3 minutes**, hovering over the page before clicking, due to confusion about the button options.
- These minor delays were attributed to unclear button labeling ("Answer 1" and "Answer 2").

Error Observations:

- **User 2** missed the Back button after completing Task 1, indicating that navigation visibility needed improvement.

- **User 2** and **User 4** hesitated during Task 2, identifying that the answer buttons were not sufficiently descriptive.

- No critical errors (complete failure to finish tasks) were observed, only minor navigational and labeling issues.

Average Completion Times:

- **Task 1:** Approximately **35–40 seconds** on average.

- **Task 2:** Approximately **2 minutes** on average.

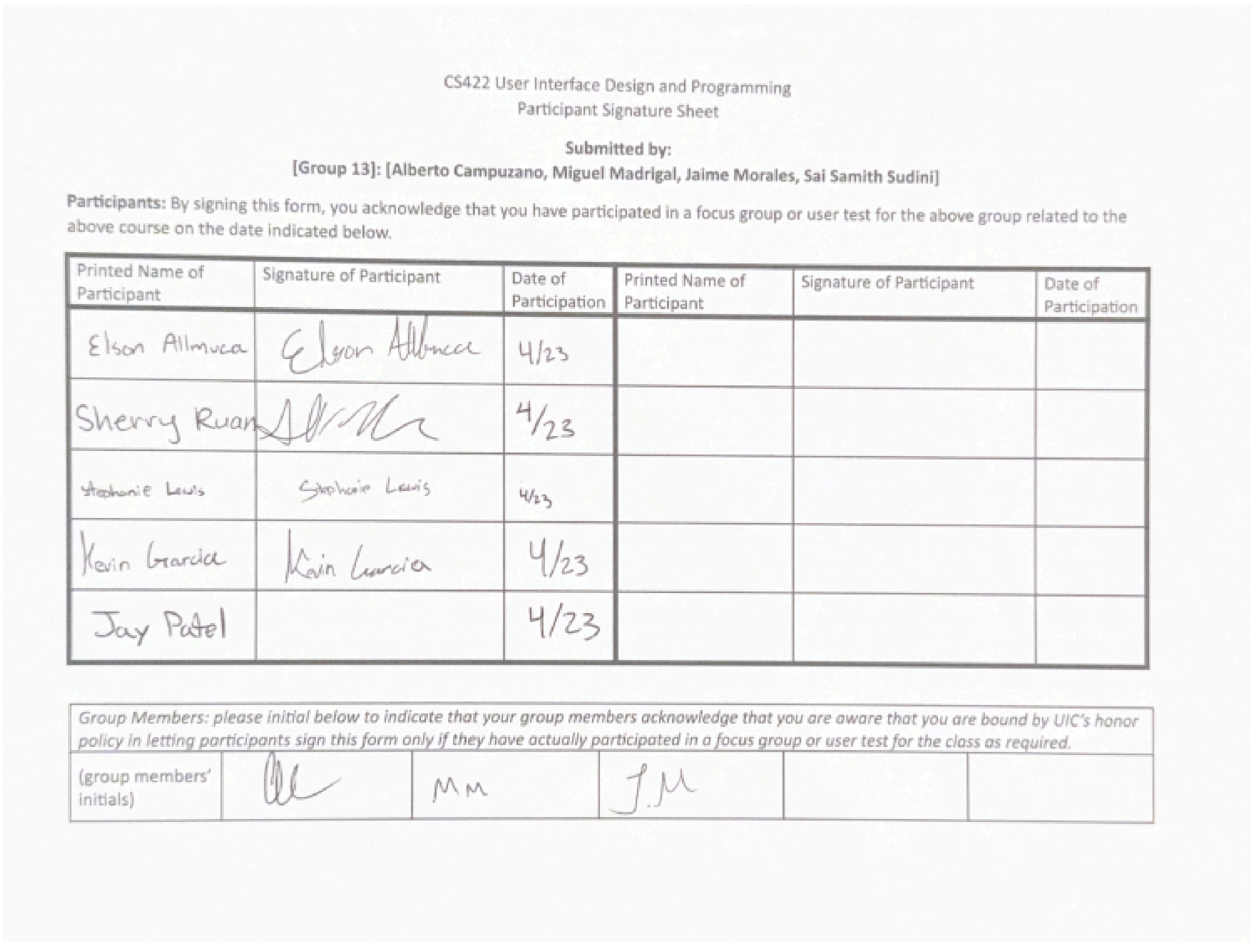
Based on user feedback and data logging results, the following improvements are recommended:

- **Improve button labeling** by replacing generic "Answer 1" and "Answer 2" with descriptive answer texts related to the lesson content.Made it simpler and less complex unless taking the quiz.

- **Enhance Back button visibility** by using a distinct color, size, or positioning to make navigation clearer.

- **Review quiz start interaction** to make sure users are confident that the quiz button is active and functional.

These changes are intended to address minor points of confusion observed during user testing and to enhance the overall usability of the CreditPath application.



CS422 User Interface Design and Programming
Participant Signature Sheet

Submitted by:
[Group 13]: [Alberto Campuzano, Miguel Madrigal, Jaime Morales, Sai Samith Sudini]

Participants: By signing this form, you acknowledge that you have participated in a focus group or user test for the above group related to the above course on the date indicated below.

Printed Name of Participant	Signature of Participant	Date of Participation	Printed Name of Participant	Signature of Participant	Date of Participation
Elson Allmuca	[illegible]	4/23			
Sherry Ruan	[illegible]	4/23			
Stephanie Lewis	Stephanie Lewis	4/23			
Kevin Garcia	Kevin Garcia	4/23			
Jay Patel		4/23			

Group Members: please initial below to indicate that your group members acknowledge that you are aware that you are bound by UIC's honor policy in letting participants sign this form only if they have actually participated in a focus group or user test for the class as required.

(group members' initials)	[illegible]	MM	J.M		

AI USAGE STATEMENT: Grammar and organization