GCIEL Assessment Strategy - R Shiny Application

Technical Documentation

Shrey Agrawal, Keiichiro Watanabe, Riku Smriga, Nandika Jhunjhunwala, Raj Jhawar

CSC- 324

Professor Priscilla Jiménez Pazmino

12/06/2023

TABLE OF CONTENTS

USER STORIES AND ACCEPTANCE CRITERIA

User Story 1: ARCS Model-Based Evaluation

As an educational researcher,

I want to use an ARCS model-based questionnaire,

So that I can evaluate the effectiveness of a VR game in terms of Attention, Relevance, Confidence, and Satisfaction.

Acceptance Criteria:

Questionnaire Accessibility:

- The app must provide a dedicated section/tab for the ARCS model-based questionnaire.
- Users should be able to access the questionnaire easily from the main interface.

Comprehensive ARCS Model Coverage:

- The questionnaire should comprehensively cover the four components of the ARCS model: Attention, Relevance, Confidence, and Satisfaction.
- Each section of the questionnaire should contain relevant questions that accurately assess its respective ARCS component.

Ease of Use and Clarity:

- The questionnaire should be easy to understand and user-friendly, with clear instructions and question wording.
- The layout and design of the questionnaire should facilitate a smooth user experience without unnecessary complexity.

Data Privacy and Security:

- The app must ensure the privacy and security of the responses collected through the questionnaire.
- Users should be informed about how their data will be used and stored.

User Story 2: Data Uploading

As a user,

I want to upload my dataset in CSV format,

So that I can analyze it using the app's features.

Acceptance Criteria:

- The app must allow the uploading of CSV files.
- It should validate the format and structure of the uploaded CSV file.
- The app should provide feedback if the uploaded file does not match the expected format.

User Story 3: Data Analysis

As a data analyst,

I want to utilize various analytical features on my dataset,

So that I can extract meaningful insights.

Acceptance Criteria:

- The app should offer features like Total Completion Time per Piece, 2D and 3D Distance Analysis, Video Engagement Analysis, and Player Positions Heatmap.
- Each feature must accurately process and visualize the data.

- The app should handle large datasets efficiently.

User Story 1: Data Export

As a researcher,

I want to download data descriptions and datasets,

So that I can use them for further analysis or reporting.

Acceptance Criteria:

- The app should provide options to download data descriptions and datasets in CSV format.
- Downloaded files should maintain data integrity and format.

PROCESS AND DEVELOPMENT FOR REPRODUCIBILITY

Data Preparation

- Data Format Specification: The app expects CSV files with specific formats. Clear guidelines are provided for users to prepare their data accordingly.

Development of Analytical Features

- Modular Design: Each feature (e.g., scatter plots, bar plots) is developed as a separate module for independent testing and updating.
- Robust Libraries: The app uses established R libraries like ggplot2 and plotly for reliable and consistent data visualization.

- Performance Optimization: Techniques for efficient data processing and minimal memory usage are implemented to handle large datasets.

User Interface and Experience

- Intuitive Design: The user interface is designed for ease of use, with clear instructions and navigation.
- Responsive Feedback: Reactive outputs provide real-time feedback as users interact with the app.
- Accessibility: The app is accessible to users with varying technical expertise.

Testing and Quality Assurance

- Unit Testing: Individual components of the app are tested to ensure they function as expected.
- Integration Testing: The app undergoes testing to ensure seamless interaction between different components.
- User Testing: Feedback from user testing sessions is used to refine usability and functionality.

Deployment

- Deployment Strategy: The app is hosted and deployed onto posit.io for ease of sharing

DESIGN DECISION DOCUMENTATION

What-Why-How Analysis on Design Decisions

UI Components

- ➢ What: The user interface is created using Shiny, featuring a title, sidebar with file upload and download links, and a main panel with tab set for different analyses.
- ➢ Why: To provide an organized and user-friendly interface for data upload, exploration, and analysis.
- ➢ How: Defined using Shiny UI functions like fluidPage, titlePanel, sidebarLayout, fileInput, downloadLink, tabsetPanel, etc.

Download Mock Data

- ➢ What: The user can choose to download mock data.
- ➢ Why: To provide sample data for the application.
- ➢ How: The user clicks on a download link, triggering a request to the server, which retrieves and provides mock data to the UI.

User Uploads CSV File

- ➢ What: User-triggered file upload event.
- ➢ Why: To allow users to use their own data.
- ➢ How: The user selects a file, and the UI sends the uploaded file to the server.

Server Processes Uploaded File

- ➢ What: The server reads the file content using read.csv.
- ➢ Why: To prepare the data for visualization.
- ➢ How: The server processes the uploaded file and stores the data in a reactive expression.

Server Renders DataTable in UI

- ➢ What: The server renders a DataTable in the UI.
- ➢ Why: To display the uploaded data for user examination.
- ➢ How: The server uses the stored data in the reactive expression to render the DataTable in the UI.

User Switches Between Tabs

- ➢ What: User-triggered tab switching event.
- ➢ Why: To navigate between different views or functionalities.
- ➢ How: The user switches tabs in the UI, triggering reactive expressions for the selected tab.

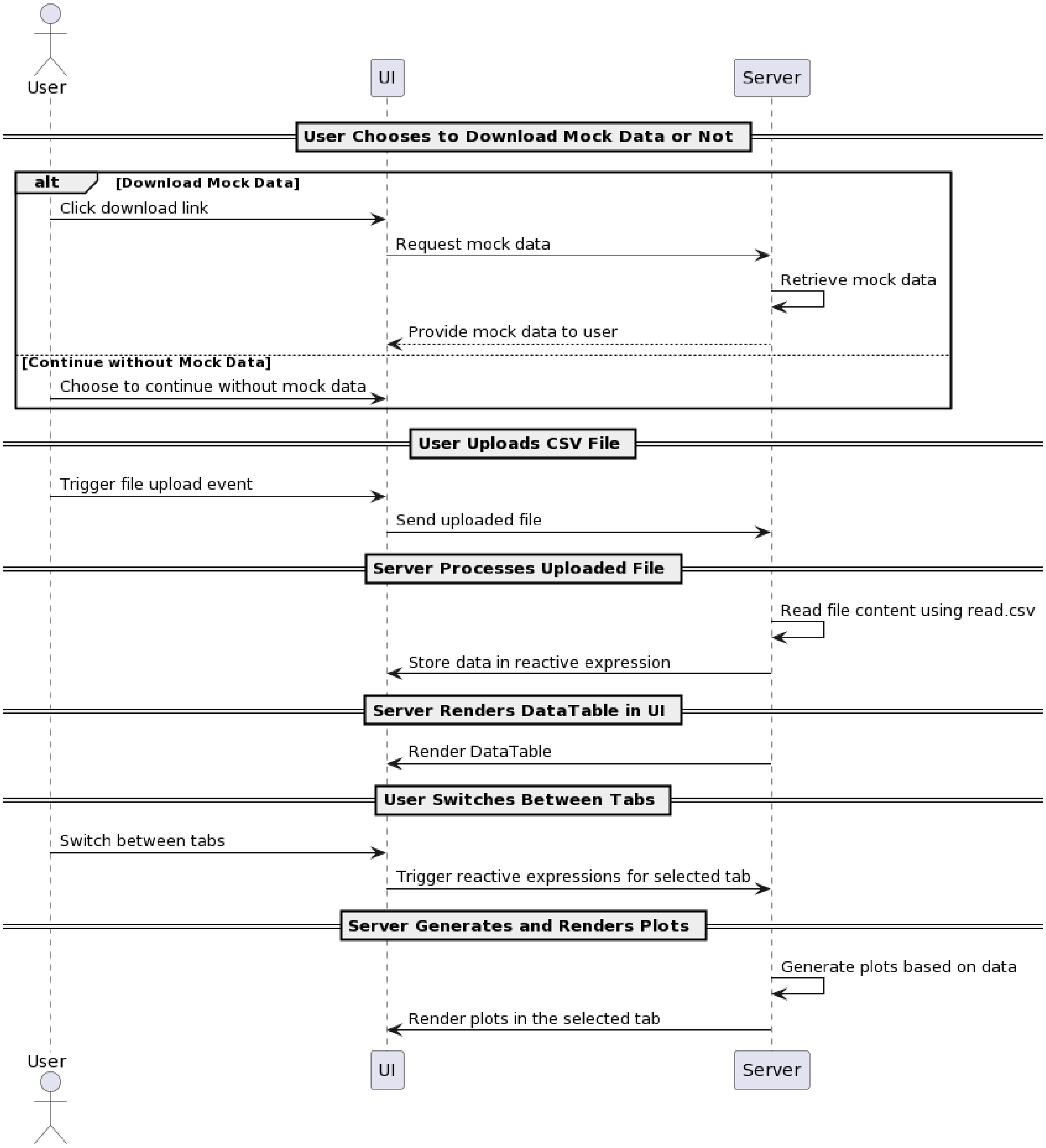
Server Generates and Renders Plots

- ➢ What: The server generates and renders plots based on the data.
- ➢ Why: To visually represent the data in different ways.
- ➢ How: The server generates plots based on the selected tab and renders them in the UI.

Google Form Integration:

- ➢ What: Embeds a Google Form in one of the tabs for user feedback or other purposes.
- ➢ Why: Collect user feedback or additional information seamlessly.
- ➢ How: Uses Shiny's renderUI to embed an HTML iframe.

UML DESIGNS


User
UI
Server
User Chooses to Download Mock Data or Not
alt
[Download Mock Data]
Click download link
Request mock data
Retrieve mock data
Provide mock data to user
[Continue without Mock Data]
Choose to continue without mock data
User Uploads CSV File
Trigger file upload event
Send uploaded file
Server Processes Uploaded File
Read file content using read.csv
Store data in reactive expression
Server Renders DataTable in UI
Render DataTable
User Switches Between Tabs
Switch between tabs
Trigger reactive expressions for selected tab
Server Generates and Renders Plots
Generate plots based on data
Render plots in the selected tab
User
UI
Server

The sequence diagram captures the flow of interactions in a Shiny web application, from the user's choice to download mock data or continue without it, to the upload of a CSV file, processing on the server, and rendering of data in the UI. It provides a visual representation of how different components of the system collaborate to fulfill user actions.

CODE CONSTRUCTION GUIDELINES (CODING STANDARDS)

Purpose

The following guidelines are designed to ensure code quality, readability, and consistency across the team's codebase. Standards apply to all R code written for the GCIEL Assessment Strategy Project

General Principles

Readability	Code should be easy to read and understand.
Consistency	Consistent style across the codebase makes the code more approachable and maintainable.
Maintainability	Code should be written with future updates and other developers in mind.

Coding Standards

1. Naming Conventions

Variables: Use clear, descriptive names. Example: shipSpeed, oceanCurrent.

Functions: Use verb-noun pairs that describe the function's action. Example: calculateDistance(), renderShip().

Constants: Use uppercase with underscores. Example: MAX_SHIP_SPEED.

2. Syntax and Style

Indentation: Use 4 spaces for indentation.

Curly Braces: Place opening braces on the same line as the function or control statement. Closing braces should be on their own line.

Line Length: Aim for a maximum of 80 characters per line for better readability.

3. Commenting and Documentation

In-line Comments: Use comments to explain complex logic or important decisions.

Function Documentation: Document the purpose, inputs, and outputs of functions.

4. Error Handling

Use consistent and robust error handling practices. Ensure that errors are caught and handled gracefully.

5. Version Control

Git: Use Git for version control.

Commit Messages: Write clear and descriptive commit messages.

6. Readability over Conciseness

Code should be easy to read and understand over being concise. Clarity should never be sacrificed for brevity.

7. White Spaces

The following are some guidelines agreed for the use of white space –

a. Use blank lines to separate groups

b. Within a block, align all the statements to the same tab stop

c. Use indentation to show the logical structure of each control structure and block

d. Use spaces around operators

8. Identifier Naming Conventions

All identifiers should be descriptive and adhering to the R programming language naming conventions. For methods, the identifier names should be verbs, using both upper and lowercase alphanumerics and with the first character of the name in lowercase.

Best Practices

Code Reviews: Regularly review code to ensure adherence to these standards.

Refactoring: Continuously improve the code for performance and readability.

Testing: Write and maintain tests for critical functionalities.

Examples

Good Practice:	**Bad Practice:**
calculateSpeed <- function(distance, time) {	*cs <- function(d, t) {*
speed <- distance / time	*s <- d / t*
return(speed)	*return(s)*
}	*}*

REFERENCES

A. Kaley, "Mapping User Stories in Agile," Nielsen Norman Group, Jan. 24, 2021. https://www.nngroup.com/articles/user-story-mapping/ (accessed Dec. 5, 2023).

B. Unhelkar, "Software Engineering with UML," 1st ed. CRC Press, 2018.

Dooley, J. F. (2017). "Software Development, Design and Coding With Patterns, Debugging, Unit Testing, and Refactoring" (2nd ed., Chapter 14). Berkeley, CA: Apress.
Available: https://doi.org/10.1007/978-1-4842-3153-1

Interaction Design Foundation - IxDF. “User Stories - Capturing the User’s Perspective Quickly and Simply” Interaction Design Foundation - IxDF. https://www.interaction-design.org/literature/article/user-stories-capturing-the-user-s-perspective-quickly-and-simply (accessed Dec. 6, 2023).

M. Domingo. “User Stories: As a [UX Designer] I want to [embrace Agile] so that [I can make my projects user-centered]” Interaction Design Foundation - IxDF. https://www.interaction-design.org/literature/article/user-stories-as-a-ux-designer-i-want-to-embrace-agile-so-that-i-can-make-my-projects-user-centered (accessed Dec. 5, 2023).

S. Sundaramoorthy, "UML Diagramming: A Case Study Approach," 1st ed. Auerbach Publications, 2022. https://doi.org/10.1201/9781003287124

S. Kurt, "Model of Motivation: ARCS Instructional Design," Education Library, Jan. 30, 2021. Updated: Oct. 17, 2022.
https://educationlibrary.org/model-of-motivation-arcs-instructional-design/ (accessed Dec. 6, 2023)

T. Munzner and E. Maguire, "Visualization Analysis & Design," Boca Raton, FL: CRC Press/Taylor & Francis Group, 2015, Ch. 3-4. https://doi.org/10.1201/b17511