

Curriculator

CEN4940 - Spring 2025

Created By:

Cole Morrison, Steven Gsell, Terra Brown, and William Money



Software to simplify course and program browsing

Project Specification

- Develop a standalone IT Program Curriculum Viewer application that allows users to query course data, identify inconsistencies, and generate reports. The application must be designed for standalone use with an embedded database or JSON for optimized data operations.

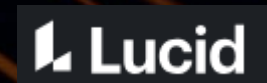
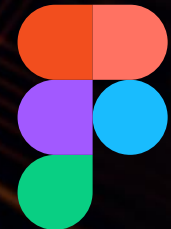
Description	Start Date	End Date	Duration
Project Start	01/19/25	04/13/25	84 Days
System Design Phase 1	1/19/25	2/2/25	14 Days
System Design Phase 2	2/2/25	2/16/25	14 Days
System Implementation Phase 1	2/16/25	3/2/25	14 Days
System Implementation Phase 2	3/2/25	3/16/25	14 Days
Integration Testing	3/16/25	4/6/25	21 Days
User Acceptance Testing	4/6/25	4/13/25	7 Days

What is Curriculator?

- A software that can accept a CSV or excel file with a specified format and displays the information to the user in a friendly navigation window. This allows easy access to:
- Courses
 - Course name / code
 - Prerequisites / Corequisites
 - The programs the course is offered in
- Programs
 - Program name / code
 - Required courses
 - Alternate courses
 - Elective courses

Technologies Used

- The project was built with several Java libraries and technology stacks
- Frontend:
 - GluonFX / JavaFX
 - CSS
 - Java
- Backend:
 - Java
 - Junit5 (testing)
 - Mockito (testing)
- Collaboration:
 - LucidChart / LucidSpark
 - Figma
 - Git / GitHub
 - Slack



Planning and Organization

- We planned out the software with user stories, class diagrams, and UI wireframing prototypes

Stack

- Java
- JavaFX/Glass
- Gradle/M
- Sublime
- Apache Commons CSV

OPTIONS TO PARSE CSV

Option	Advantages	Disadvantages
Apache Commons CSV	Simple to use, well-documented	Requires external dependency
OpenCSV	Simple to use, no external dependencies	Less feature-rich than Commons CSV
Handwritten parser	Full control, no dependencies	Time-consuming to develop and maintain

Completed Work

Task	Progress	Start Date	Due Date	Assignee
Project Setup	Completed	2023-01-01	2023-01-05	John
Requirement Gathering	Completed	2023-01-06	2023-01-10	John
UI Wireframing	Completed	2023-01-11	2023-01-15	John
Class Diagramming	Completed	2023-01-16	2023-01-20	John
Database Design	Completed	2023-01-21	2023-01-25	John
Backend Logic Development	In Progress	2023-01-26	2023-02-10	John
Frontend Logic Development	In Progress	2023-01-26	2023-02-10	John
Integration Testing	Not Started	2023-02-11	2023-02-20	John
Deployment	Not Started	2023-02-21	2023-03-01	John

Current Work

Task	Progress	Start Date	Due Date	Assignee
Backend Logic Development	In Progress	2023-01-26	2023-02-10	John
Frontend Logic Development	In Progress	2023-01-26	2023-02-10	John
Integration Testing	Not Started	2023-02-11	2023-02-20	John
Deployment	Not Started	2023-02-21	2023-03-01	John

Extra Notes

Integration testing is a critical part of the development process. It ensures that the different components of the application work together as expected.

hasCSV V2

noCSV

hasCSV

changeCSV

programDetail

courseDetail

Program Operation

- Turning excel sheets that make your head spin into easily readable data!

	A	B	C	D	E	F
1	COURSE	COURSE NAME	PREREQUISITES	300-Computer Systems Network	3301-Information Systems Technology-Application Developm	301-Information Systems Technology-Fin
2	CAP2140	Data Forensics I	CT31131C and CT31133C and CT31120C			
3	CAP2141	Data Forensics II	CAP2140			
4	CAP2741C	Data Visualization	COP2822C and COP2034C and CGS2512C			
5	CAP2787C	Data Warehousing	CT32437C			
6	CEN2071C	Introduction to Software Testing	COP2551C or COP2334C or COP2360C or COP2837C or COP2220C or COP2800C or COP		R	
7	CEN3024C	Software Development I	COP3330C		R	
8	CEN3083C	Introduction to Cloud Computing	CNT2001C or CET2600C		R	
9	CEN4025C	Software Development II	CEN3024C co CEN4502C		R	
10	CEN4802C	Software Integration, Configuration, and Testing	CEN3024C and CEN2071C co CEN4025C		R	
11	CEN4340	Application Development Capstone	CEN4025C and COP4655C and CEN4802C and COP4847C		R	
12	CET2600C	Network Fundamentals (Cisco 1)	CT31131C and CT31133C		A	
13	CET2662	Techniques of Computer Hacking and Incident Handling	CT31131C and CT31133C and CT31120C			A
14	CGS1060C	Introduction to Information Technology				
15	CGS1100C	Microcomputer Applications for Business and Economics			R	
16	CGS2512C	Spreadsheet Concepts and Practices				R
17	CGS2542	Database Concepts for Microcomputers	CGS1060C or CGS1100C			
18	CGS2820	Web Site Design and Development	COP2822C			
19	CGS2821	Advanced Web Site Design and Development	CGS2820			
20	CI3184Z	Computer Information Technology Internship				
21	CI3232I	Information Systems	CGS1100C		R	
22	CI32349C	Introduction to Big Data using Hadoop	COP2034C and COP2800C and CT32437C and CNT1015C			R
23	CI32930	Computer Information Technology Special Topics				
24	CI33534C	Scripting for Network Professionals	CT31133C and CET2600C and CT32655C	R		
25	CNT1015	Operating Systems Foundations		R		
26	CNT2001C	Computer Networks and Telecommunications	CGS1060C or CGS1100C		R	
27	CNT2102C	Enterprise Networking, Security, and Automation (Cisco 3)	CT32655C	R		
28	CNT2404	Intrusion Detection Systems and Auditing	CET2662			
29	CNT2942	IT System Technology & IT Security Cooperative Education (Internship)				
30	CNT3014C	Enterprise Systems Integration	CT32655C	R		
31	CNT3105C	Software Defined Networking	CT31534C and CNT2102C and CI33534C	R		
32	CNT3401C	Firewall Configuration and Design	CNT2102C and CNT4108C and CNT3105C	R		
33	CNT4509C	Advanced Convergent Technologies	CNT2102C	R		
34	CNT4108C	Advanced Network Traffic Analysis	CT32655C	R		
35	CNT4331C	Computer Networking Capstone	CNT3014C and CNT4509C	R		
36	CNT4340	Computer Systems Networking Cooperative Education (Internship)	ISM3013 and ISM3014	R		
37	COP1000C	Introduction to Computer Programming			R	
38	COP2034C	Programming in Python	COP1000C			R
39	COP2073C	Introduction to Statistical Programming with R	COP1000C			
40	COP2220C	C Programming	COP1000C			
41	COP2334C	Object-Oriented Programming with C++	COP1000C			
42	COP2360C	Introduction to C#	COP1000C		A	A
43	COP2551C	Introduction to Object Oriented Programming with Java	COP1000C		R	R
44	COP2800C	Java I	COP1000C		A	A
45	COP2805C	Advanced Java Programming	COP2551C or COP2800C			
46	COP2806C	Developing Enterprise Applications Using Java EE	COP2805C			
47	COP2822C	Web Technologies	COP1000C			
48	COP2823C	ASP.NET Programming	COP2837C or COP2360C			
49	COP2837C	Introduction to Programming with Visual Basic.NET	COP1000C			
50	COP2842C	Internet Programming	COP2822C		A	A
51	COP3330C	Object-Oriented Programming	COP2551C or COP2800C or COP2360C		R	
52	COP3815C	Web Application Development			R	
53	COP4655C	Application Development for Mobile Devices	CT32437C and COP3330C and CEN2071C and ISM3113C		R	
54	COP4847C	Advanced Web Application Development	COP3330C and COP3815C		R	
55	CT31120C	Fundamentals of Information Security	CT31131C and CT31133C or CGS1060C and CNT1015C and CNT2001C	R		R
56	CT31131C	Software Configuration			R	R
57	CT31133C	Hardware Configuration			R	R
58	CT31136	A+ Certification Review				
59	CT31154	Technical Support		R		
60	CT31534C	Server Configuration	CT31131C and CT31133C	R		



Curriculator

HomeChange CSV

Search

File loaded: ITPROGRAMS.csv

Courses

Programs

CAP2140 - Data Forensics I

CAP2141 - Data Forensics II

CAP2741C - Data Visualization

CAP2787C - Data Warehousing

CEN2071C - Introduction to Software Testing

CEN3024C - Software Development I

CEN3083C - Introduction to Cloud Computing

CEN4025C - Software Development II

CEN4802C - Software Integration, Configuration, and Testing

CEN4940 - Application Development Capstone

CET2600C - Network Fundamentals (Cisco 1)

CET2662 - Techniques of Computer Hacking and Incident Handling

CGS1060C - Introduction to Information Technology

CGS1100C - Microcomputer Applications for Business and Economics

CGS2512C - Spreadsheet Concepts and Practices

CGS2542 - Database Concepts for Microcomputers

CGS2820 - Web Site Design and Development

CGS2821 - Advanced Web Site Design and Development

CI3184Z - Computer Information Technology Internship

CI3232I - Information Systems

CI32349C - Introduction to Big Data using Hadoop

CI32930 - Computer Information Technology Special Topics

CI33534C - Scripting for Network Professionals

CNT1015 - Operating Systems Foundations

CNT2001C - Computer Networks and Telecommunications

CNT2102C - Enterprise Networking, Security, and Automation (Cisco 3)

CNT2404 - Intrusion Detection Systems and Auditing

CNT2942 - IT System Technology & IT Security Cooperative Education (Internship)

CNT3014C - Enterprise Systems Integration

CNT3105C - Software Defined Networking

CNT3401C - Firewall Configuration and Design

CNT4509C - Advanced Convergent Technologies

CNT4108C - Advanced Network Traffic Analysis

CNT4331C - Computer Networking Capstone

CNT4340 - Computer Systems Networking Cooperative Education (Internship)

COP1000C - Introduction to Computer Programming

COP2034C - Programming in Python

COP2073C - Introduction to Statistical Programming with R

COP2220C - C Programming

COP2334C - Object-Oriented Programming with C++

COP2360C - Introduction to C#

COP2551C - Introduction to Object Oriented Programming with Java

COP2800C - Java I

COP2805C - Advanced Java Programming

COP2806C - Developing Enterprise Applications Using Java EE

COP2822C - Web Technologies

COP2823C - ASP.NET Programming

COP2837C - Introduction to Programming with Visual Basic.NET

COP2842C - Internet Programming

COP3330C - Object-Oriented Programming

COP3815C - Web Application Development

COP4655C - Application Development for Mobile Devices

COP4847C - Advanced Web Application Development

CT31120C - Fundamentals of Information Security

CT31131C - Software Configuration

CT31133C - Hardware Configuration

CT31136 - A+ Certification Review

CT31154 - Technical Support

CT31534C - Server Configuration

S300 - Computer Systems Networking

S301 - Information Systems Technology

2153 - Computer Information Technology

2156 - Network System Technology

2157 - Data Science Technology

2158 - IT Security

4301 - Financial Technology

6109 - Network Support Technician

6110 - Network Infrastructure

6111 - Advanced Network Virtualization

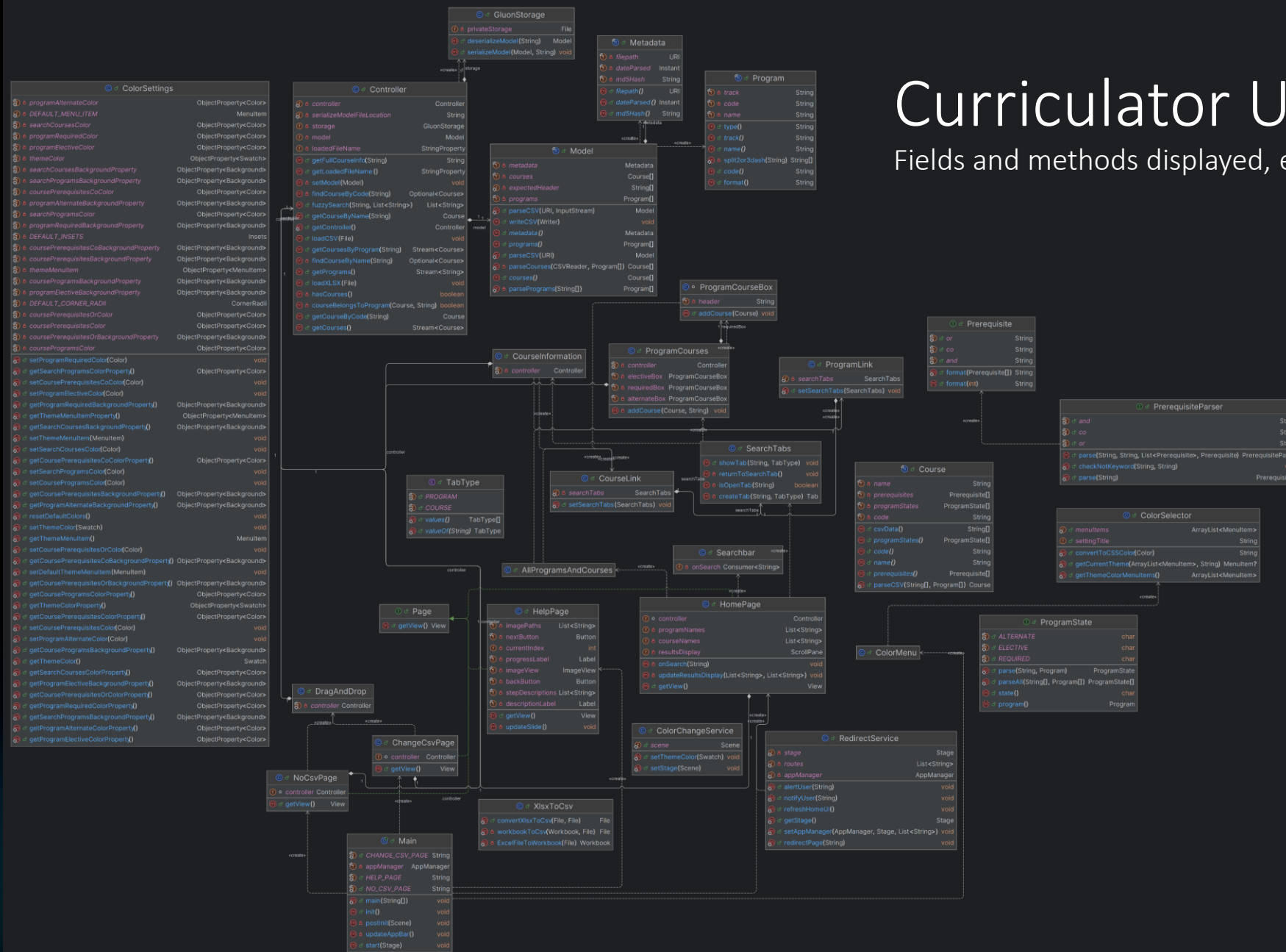
6113 - Network Enterprise Administration

6114 - IP Communications

6999 - Information Technology Career Pathways

Curriculator UML Diagram

Fields and methods displayed, excluding test packages



Code Structure

- Used MVC pattern for our code structure:
- Model – Manages the data
- View – The user interface built with JavaFX
- Controller – Connects the view and model, handles business logic

📁 controller

📁 frontend

📁 model

📄 Main.java

The Model

- Defines core data structures such as Courses, Programs, and Metadata.
- Provides methods for the Controller to parse a CSV file.
- Uses Java records and sealed interfaces for clear, type-safe parsing.
- Thoroughly unit tested, 100% code coverage!

▼ model

- 📄 ColorSettings.java
- 📄 Course.java
- 📄 Metadata.java
- 📄 Model.java
- 📄 Prerequisite.java
- 📄 PrerequisiteParser.java
- 📄 Program.java
- 📄 ProgramState.java

Model - ProgramState

- Represents an association between a Program and Course.
- Each Course may be Required, Alternate, or Elective in each Program.
- Uses the Program parsed from the CSV header, encapsulating it in a sealed interface for type-safe handling later.

```
/**
 * Returns the program associated with this state. Will be automatically
 * implemented by the record classes, which contain Program program().
 *
 * @return the program associated with this state
 */
Program program();

/**
 * Represents a required program state.
 *
 * @param program the program associated with this state
 */
public static record Required(Program program) implements ProgramState {

/**
 * Represents an alternate program state.
 *
 * @param program the program associated with this state
 */
public static record Alternate(Program program) implements ProgramState {

/**
 * Represents an elective program state.
 *
 * @param program the program associated with this state
 */
public static record Elective(Program program) implements ProgramState {
```

Model - PrerequisiteParser

- Parses Prerequisite Course codes using a state machine.
- Prerequisites separated by “and” must all be satisfied.
- A group of Prerequisites separated by “or” are alternates, separate from other Prerequisites listed.
- A Corequisite is indicated by “co” and a Course code.

“COP1000 and COP2000 or COP3000 co COP4000” →

Prerequisites: COP1000, (COP2000 or COP3000)

Corequisites: COP4000

```
/**
 * Parse method, which each state implements differently.
 */
⌘Cody
PrerequisiteParser parse(String prerequisites, String token,
    List<Prerequisite> results, Prerequisite last) throws ParseException;

⌘Cody
> static record And() implements PrerequisiteParser { ...

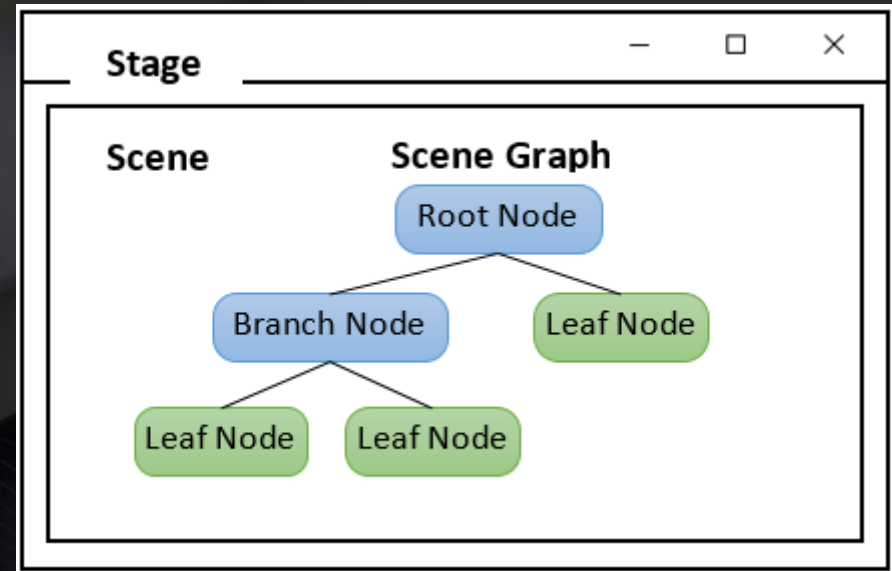
⌘Cody
> static record Or() implements PrerequisiteParser { ...

⌘Cody
> static record Corequisite() implements PrerequisiteParser { ...

⌘Cody
> static record Keyword() implements PrerequisiteParser { ...
```

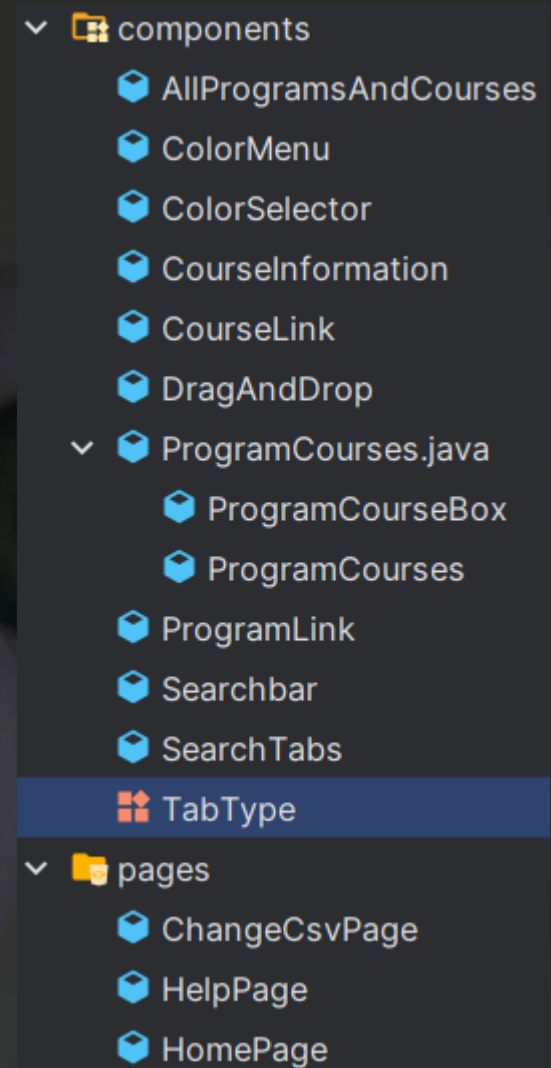

The View

- Built using JavaFX
- Responsible for what the user sees and interacts with
- Uses a Stage (main application window)
- Displays content through Scenes
- Contains UI components like buttons, text fields
- Sends user actions (like clicks or input) to the Controller



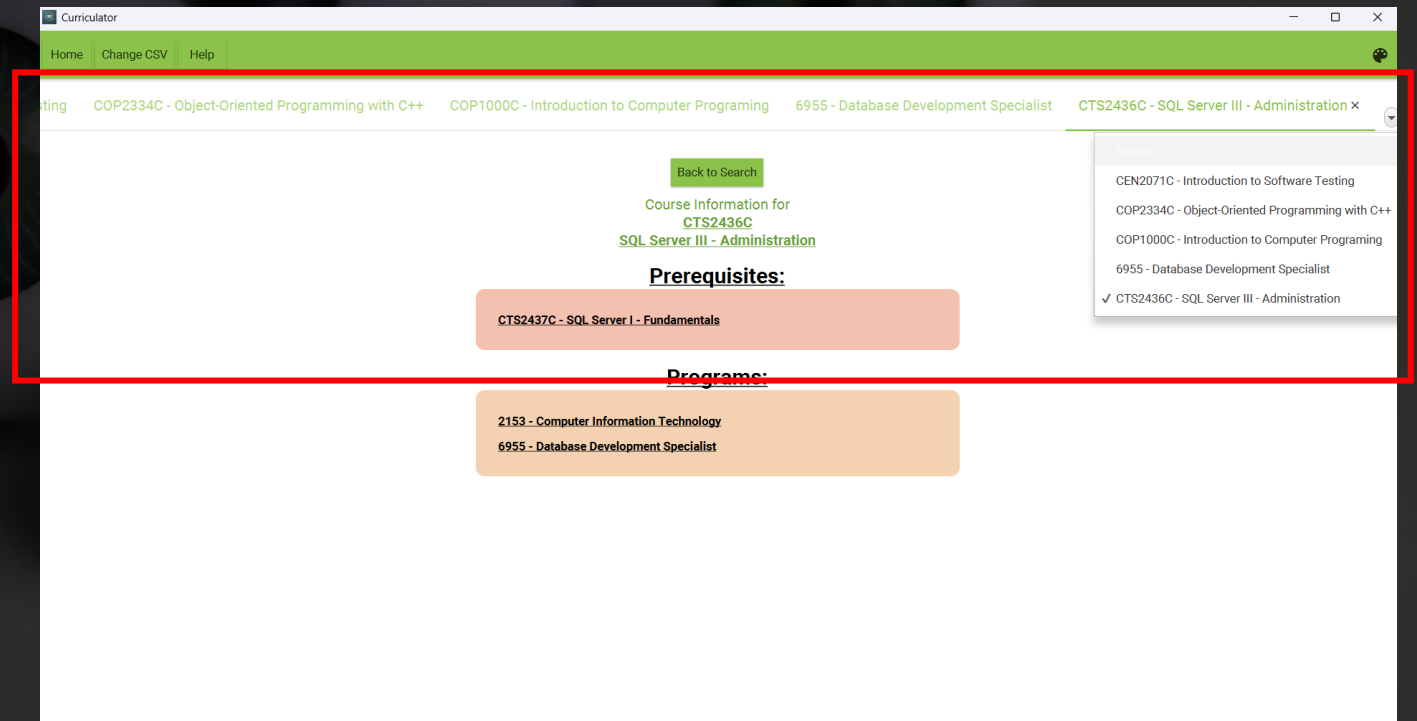
View Components

- Although the UI is simple there a lot of inner parts!
 - A dozen components
 - Three pages (scenes)
 - A service to aid page redirection
 - A service to aid color options



View Tab System

- With collaboration and planning we designed a tab system
 - Allows user to “drill down” into programs and course prerequisites
 - When the tabs overflow a simple dropdown list allows easy navigation



The Controller

A bridge between the model and the view

- Utilizes the Singleton Pattern to ensure consistent access to data across the views
- Loads, parses, and serializes the model
- Feeds the view, supplying lists of courses and programs for display in the UI
- Handles the data flow, when a user uploads or searches data, the view talks to the controller, and the controller talks to the model.

```
Controller.java x
36  * program data from the model in a stream-oriented and JavaFX-friendly way.
37  */
38  public class Controller { 30 usages 1 moneymat7 +3
39
40      // Path to the serialized model file
41      private static String serializeModelFileLocation = "../SerializedCurriculatorModel.ser"; 2 usages
42
43      // Singleton instance of Controller
44      private static Controller controller = new Controller(); 3 usages
45
46      // JavaFX property to store the loaded CSV file name or status
47      private StringProperty loadedFileName = new SimpleStringProperty("File not found."); 5 usages
48
49      /**
50       * Returns the JavaFX property representing the currently loaded CSV filename.
51       * Can be bound to UI components.
52       *
53       * @return the {@link StringProperty} representing the file name status.
54       */
55      public StringProperty getLoadedFileName() { return loadedFileName; }
56
57      /**
58       * Returns the singleton instance of the {@code Controller}.
59       *
60       * @return the singleton {@code Controller} instance.
61       */
62      public static Controller getController() { 1 moneymat7 +1
63          if (controller == null)
64              controller = new Controller();
65
66          return controller;
67      }
68
69      // The application's course and program data model
70      private Model model; 12 usages
71
72      /**
73       * Sets the current model and updates the loaded file name property.
74       *
75       * @param model the {@link Model} to set, or {@code null} to clear the model
76       */
77      private void setModel(Model model) { 2 usages 1 Terra Brown +1
78          this.model = model;
79      }
80  }
```

Controller - Storing Application Data

- Users upload CSV or XLSX files into the application
- The application converts these files into Java objects
- Java objects run in memory and are lost when the app closes
- To retain data, Java objects are serialized into a .ser file
- The serialized file is saved for future retrieval and use

Controller - Gluon Storage

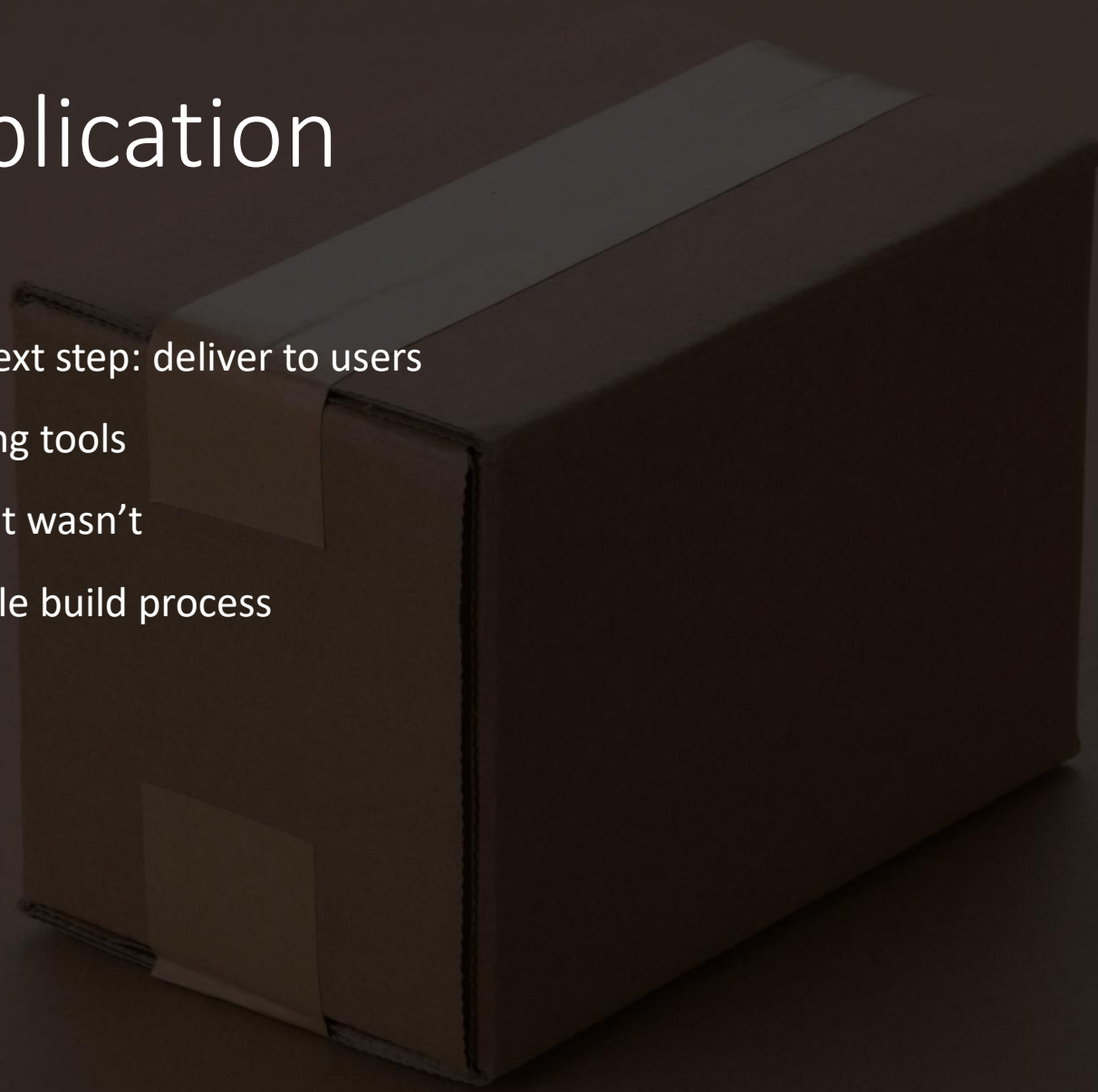
- The application uses the Gluon framework, which provides a built-in storage service
- Gluon Storage offers several benefits over standard file storage:
 - Stores files in a private location accessible only to the app
 - Ensures cross-platform compatibility by finding the recommended storage location per platform
 - Automatically deletes files when the application is uninstalled
 - Offers a secure, platform-agnostic method for storing application data

Application Logging

- Use Log4j2 for flexible and high-performance logging
- Configure logger settings using an XML file
- Define log output file in the configuration
- Instantiate logger in each class
- Use logger to add log statements across the application

Packaging Application

- App worked in development next step: deliver to users
- Gluon provides native packaging tools
- Expected it to be simple... but it wasn't
- Needed a consistent, repeatable build process



Packaging – Local Issues

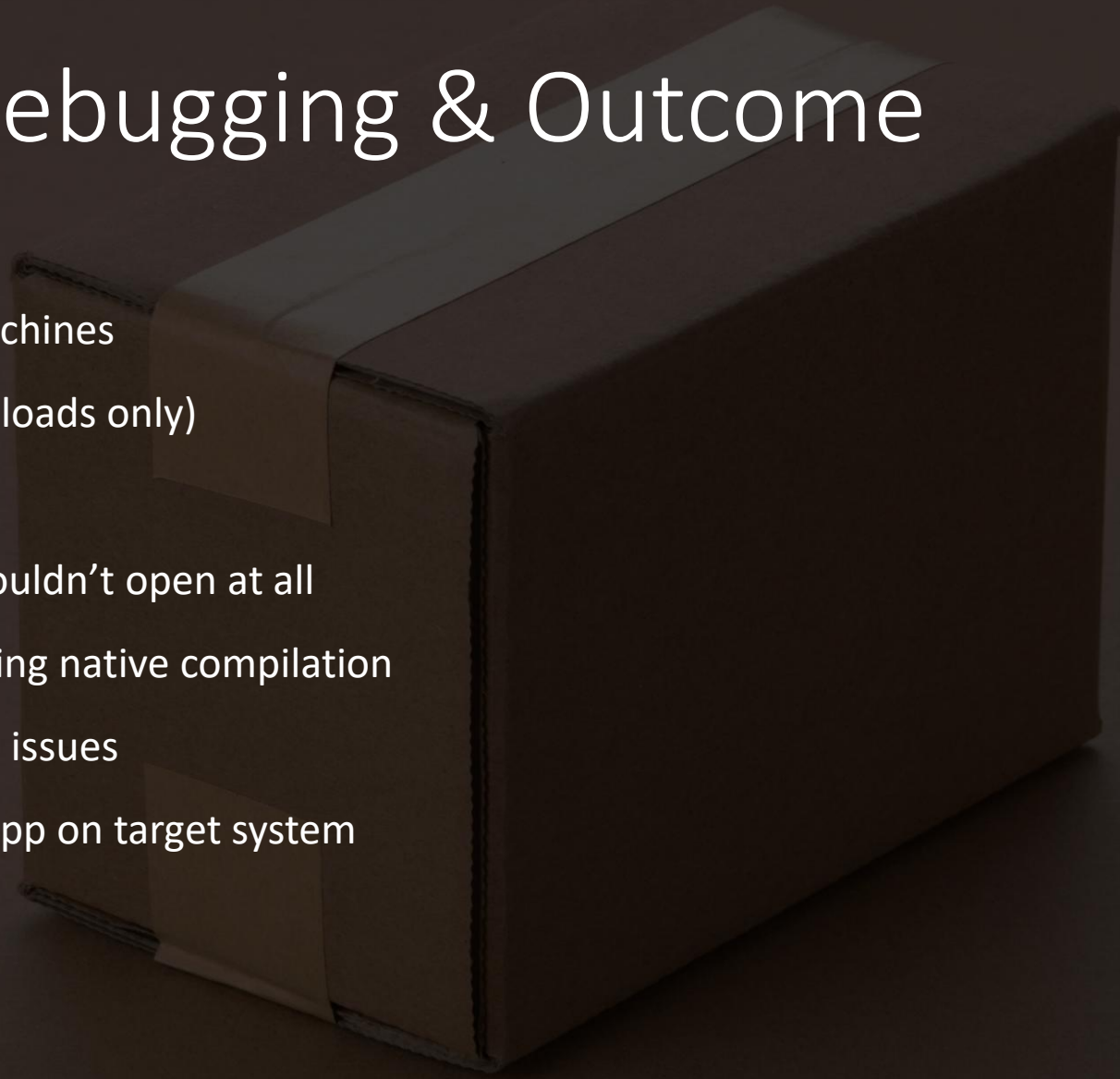
- Ran into dependency problems
- Required many tools to be manually installed
- Local setup was fragile and time-consuming
- Decided to switch to a clean, isolated build environment

Packaging – Using GitHub Actions

- GitHub Actions = Cloud servers for automation
- Workflow is defined in a .yaml file inside the repository
- Workflow setup:
 - Pull code → Install dependencies → Package → Create Artifact
- Artifacts generated and downloadable directly from GitHub

Packaging – Debugging & Outcome

- .exe behavior varied across machines
- Worked on Horizon PC (CSV uploads only)
- Failed on some other PCs
- Added SLF4J logging → app wouldn't open at all
- Likely dependency conflict during native compilation
- Tried JVM packaging, got same issues
- Final result: partially working app on target system



Problems We Faced



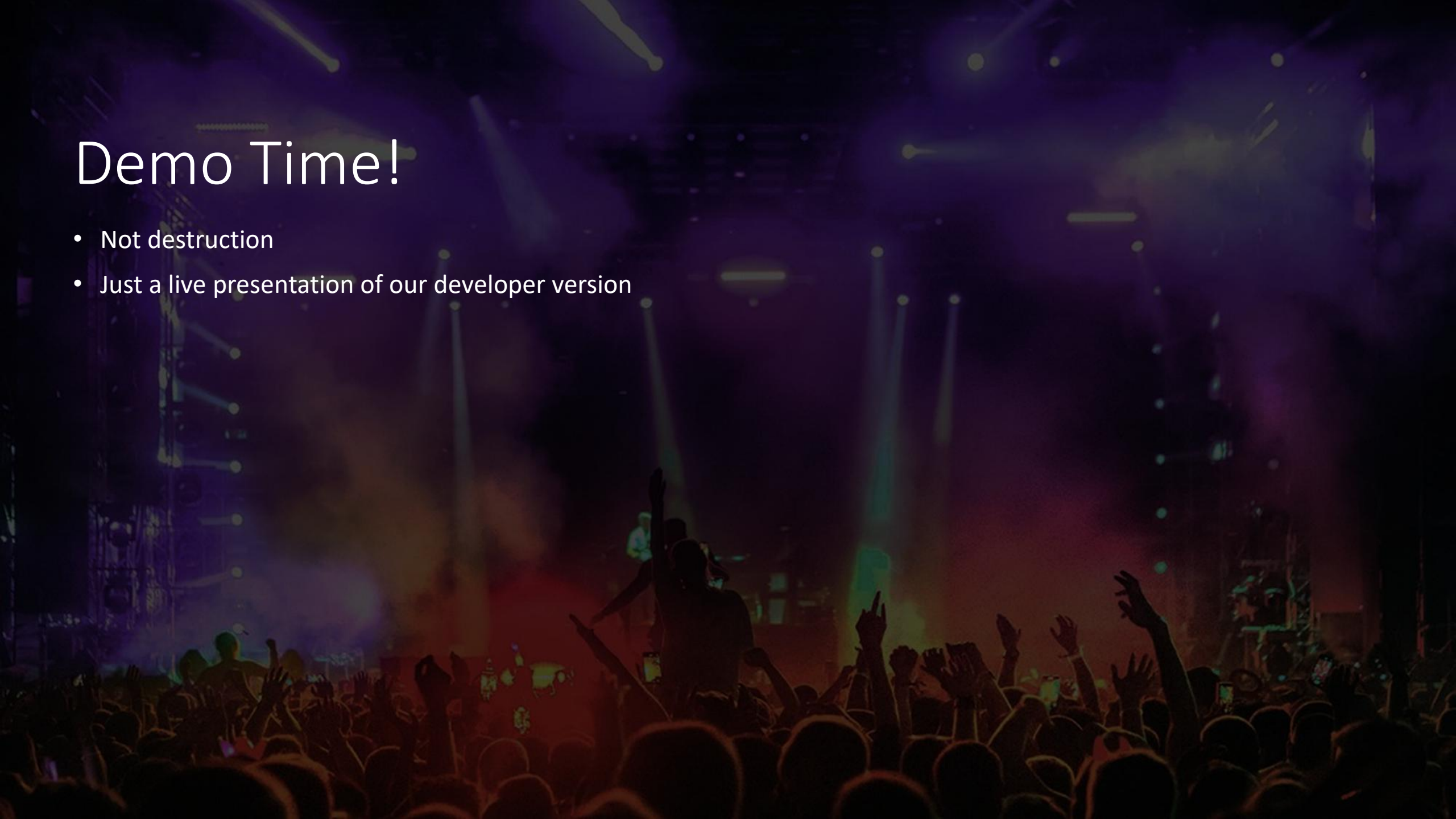
- What technologies to use
- How to divide up the work
- How to work cohesively without creating merge conflicts (more than one person editing a file)
- How to fix bugs that exist only in the compiled version
- Figuring out the point to stop adding non-functional features

The Backlog

- Exporting to a formatted file
 - Request of JSON format
- CeL Master sheet implementation
 - Status
 - Updated dates
- Compiled Bugs
 - Drag and Drop functionality
 - Serialization / Deserialization
- Dynamic Loading
 - Load data on original file change

Demo Time!

- Not destruction
- Just a live presentation of our developer version



Conclusion

- This course has been the pinnacle of all that we have learned here at FSCJ
- It was so much fun to collaborate and work our way through developing functional software from scratch!
- We will take what we have learned here to make the software of the future
- Thanks for coming!