MICHIGAN STATE UNIVERSITY

Project Plan Presentation

FGL Parser & Renderer

The Capstone Experience

Team Ludus

Abhay Saji
Umut Temel
Nicholas Seals
Isabella Nelsen
David Oh
Zakariya Sattar

Department of Computer Science and Engineering
Michigan State University

Fall 2025

From Students…
…to Professionals

Project Sponsor Overview

- Founded in 2016 - Kevin Schneider/Zachary Collins
- Used by 3500+ Organizations
- Ticketing Service for events
- Inc. 5000 (2023-2024)

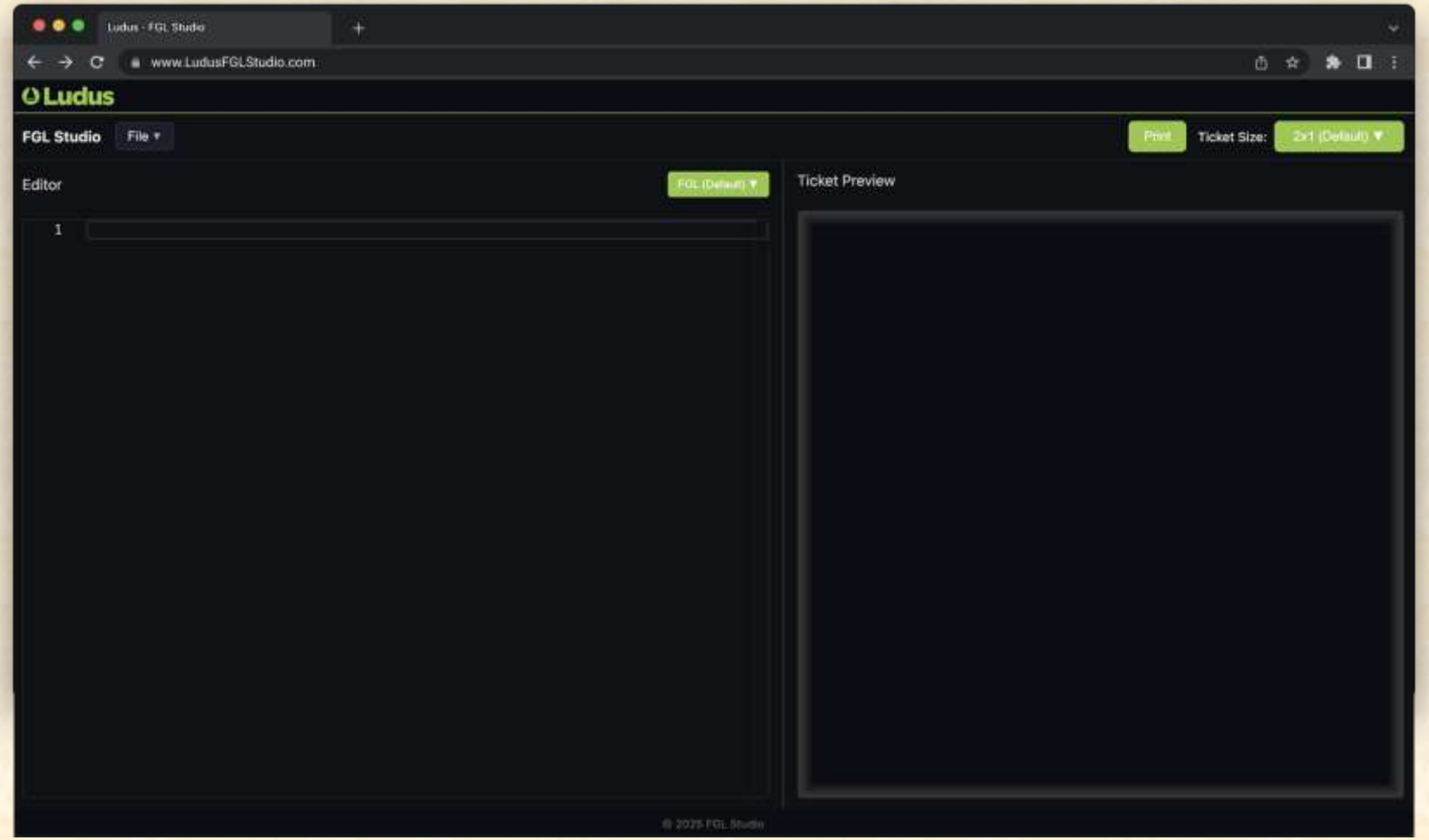
Project Functional Specifications

- Simplify the creation of BOCA tickets
- By creating a web-based editor
- With reality-accurate rendering
- And modern IDE features

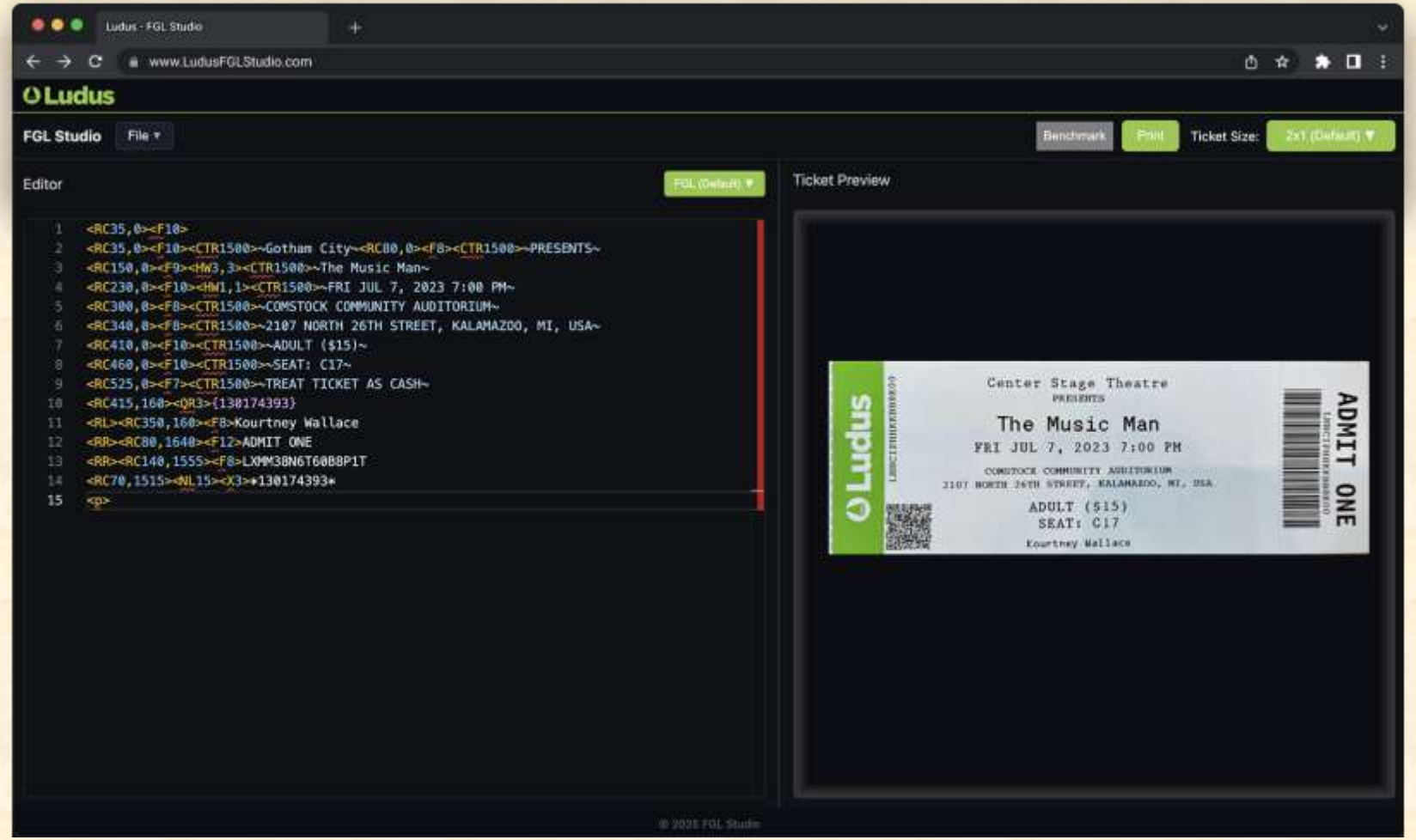
Project Design Specifications

- FGL text editor with text highlighting, text autocompletion, and error reporting
- Ticket renderer that responds to edits in the given FGL code
- Ability to output straight from website to printer
- File explorer and database system to import code

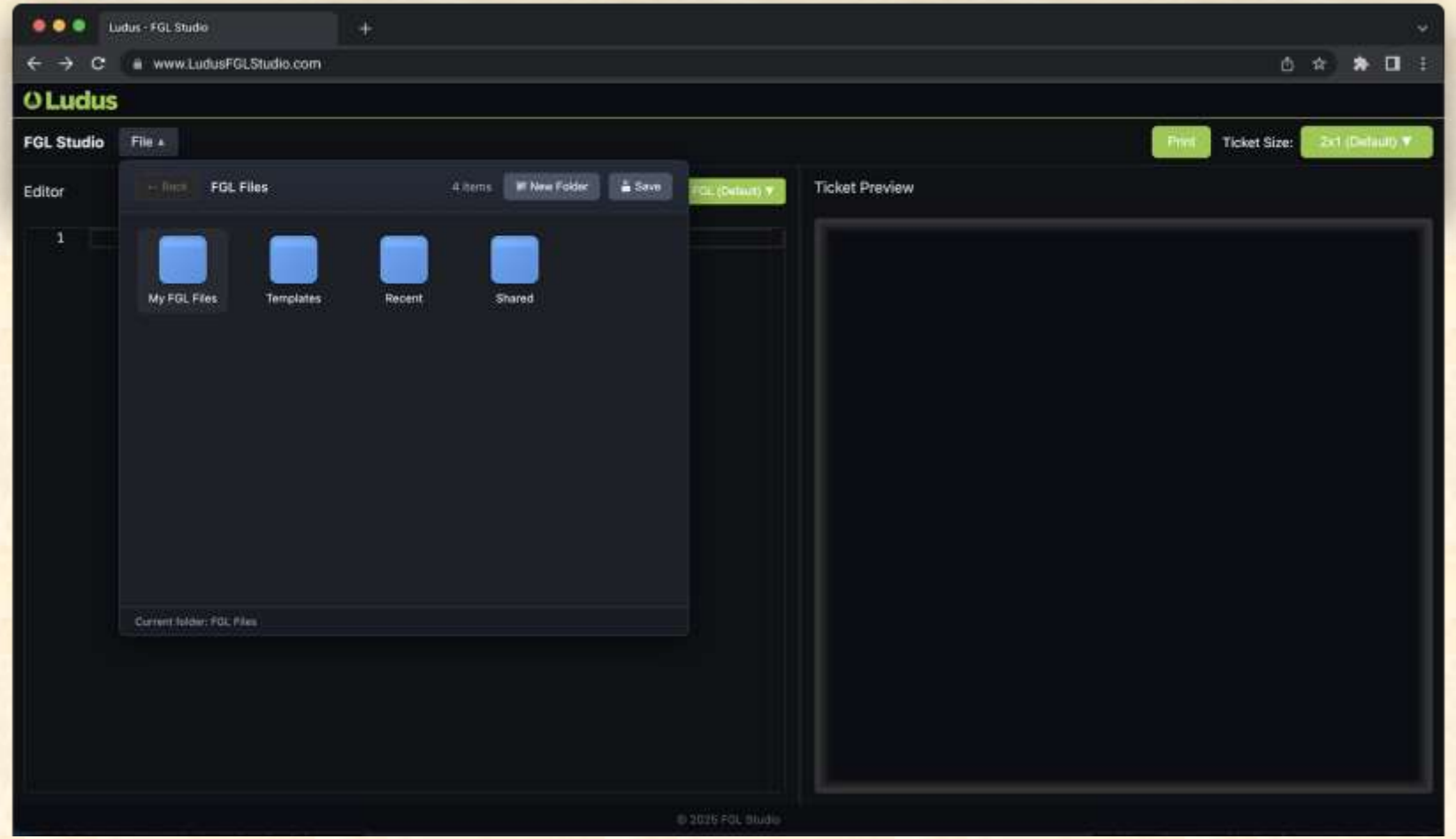
Screen Mockup: General Page



Screen Mockup: Ticket Preview



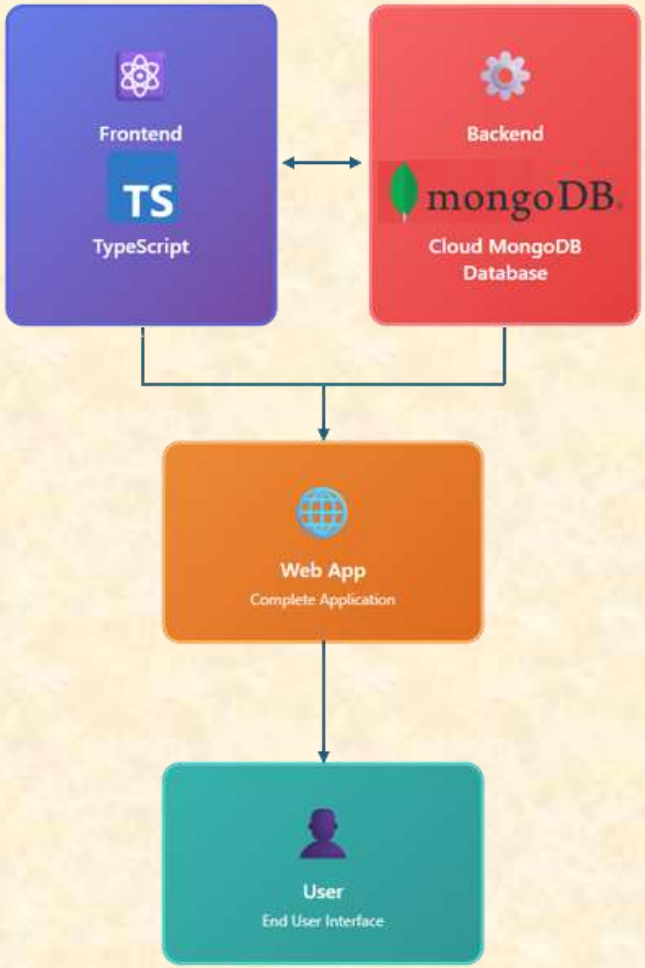
Screen Mockup: File Menu



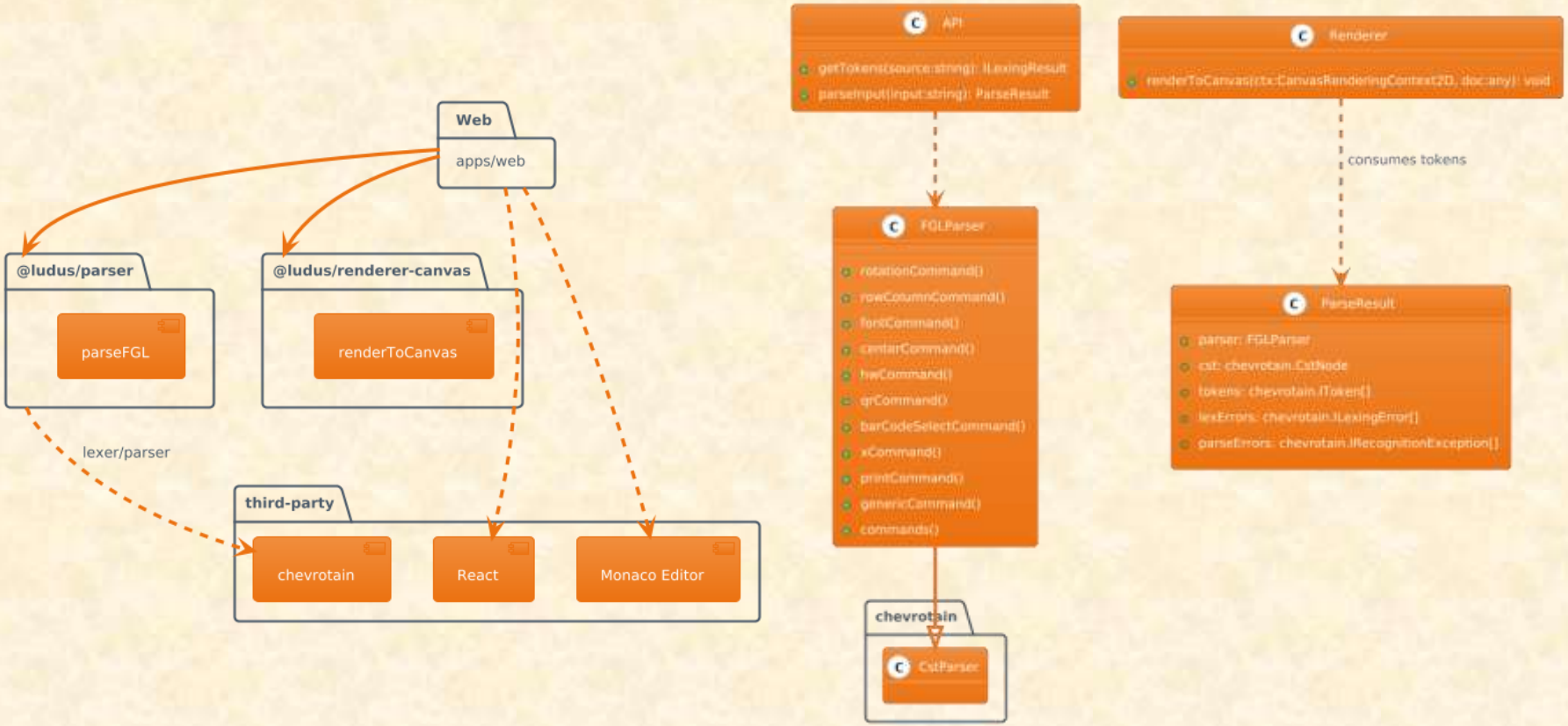
Project Technical Specifications

- TypeScript and React frontend
- MongoDB as Database
- Chevrotain parser generator for FGL syntax analysis
- HTML5 Canvas API for rendering

Project System Architecture



Project System Architecture (UML)



Project System Components

- Hardware Platforms
  - BOCA Printer supplied by Ludus for testing
- Software Platforms / Technologies
  - MongoDB Database
  - Installed Typescript as a development dependency in our project
  - Actively using the Chevrotain and Monaco Editor libraries in building our application
  - Custom Renderer

Project Risks

- Risk 1 - Accurate Barcode Rendering and Visualization
  - Rendering scannable barcodes is a necessity for creating a viable and usable printing emulation service. Minor scaling or DPI difference can completely break scanning and formatting, and moreover nobody on the team has worked with barcodes thus far.
  - Attempt to stick with well-maintained barcode libraries instead of hand rolling features, with a strong scanner-based testing suite built into our CI pipeline to ensure we don't break this inevitably fragile part of our code base.
- Risk 2 - Building a Lexer To Efficiently Parse FGL
  - It is difficult to build a parser/lexer that is efficient for the FGL language, it is unknown to us how to convert the FGL language into objects that can be rendered.
  - Start reading and learning about how languages like HTML parse their language and implement a similar logic for FGL. We'll need to implement extensive unit tests for each command, as part of not only documenting the code but preventing regressions as we build out the parser.
- Risk 3 - Automate Comparison of Rendering Output to Physical Printing Output
  - It is difficult to verify that what we render is going to be printed by the printer programmatically, as we want to ensure that our render is perfect down to the dot.
  - Print tickets as an immediate stop-gap, and then further examining previous software renders and deep-diving more into testing methods as a spike for this project. Look into methods such as digitally scanning and processing tickets to automatically compare them to our digital renders.
- Risk 4 - Specification Ambiguity & Partial Command Support
  - Boca's official FGL documentation, as mentioned in the project proposal can be sparse regarding some commands. While the proposal starts off requesting a subset of commands, Ludus may expect more over time leading to scope creep as well.
  - A well-formed parser to make supporting new commands simpler to parse will be key. Following that by developing a comprehensive unit test suite for each command we can add test cases that would cover edge cases, making development much easier.