Language Learning Adventure

An Interactive Functional Programming Project

Adan Fhima, Dan Suissa, Karel Moryoussef and Maximilien Bruck

Introduction & Objectives

Project Overview:

- *Language Learning Adventure*: An interactive language learning application using a tree structure.
- Developed using **Haskell** in the context of a **Functional Programming** course.

Purpose & Objectives:

- Apply functional programming principles in a practical project.
- Create an engaging, effective language learning tool.
- Utilize advanced Haskell features (custom data types, monads, parser combinators).

Agenda:

Project Overview

Description:

- *An interactive language learning application supporting:*
  - **Spanish -**
  - **French**
  - **German**

```
Language Learning Adventure
|
├── Language: Spanish
|   └── Levels → Lessons → Content
|
├── Language: French
|   └── Levels → Lessons → Content
|
└── Language: German
    └── Levels → Lessons → Content
```

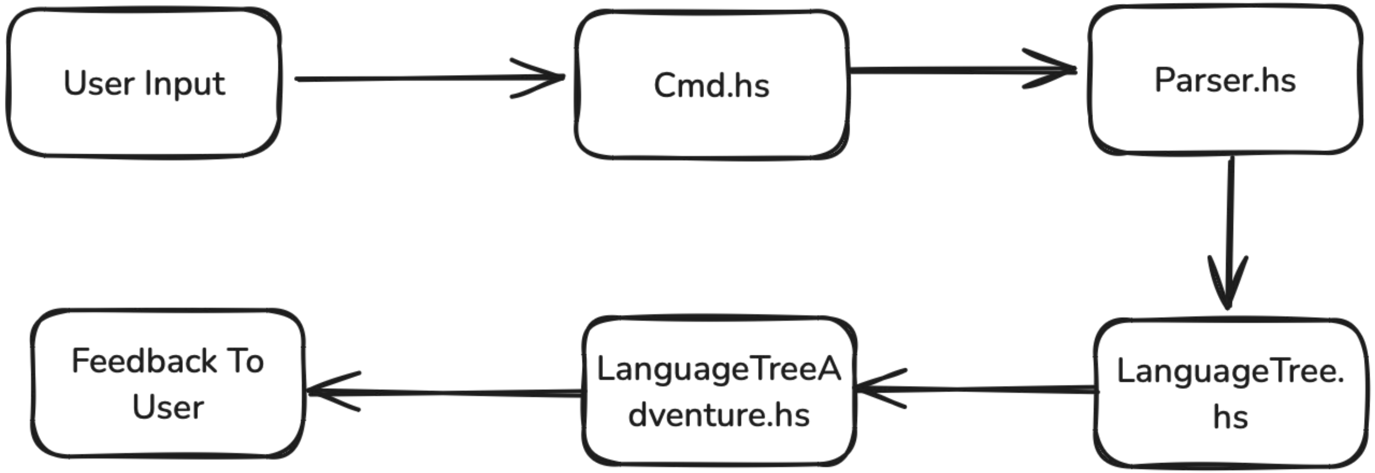
Goals:

- **Structured Lessons:**
  - Vocabulary
  - Grammar explanations
  - Quizzes for reinforcement

- **User Progress Tracking:**
  - Monitor completed lessons and levels
  - Provide feedback and scores

Code Architecture and Game Flow

- The game consists of 4 main files - Cmd.hs, Parser.hs, LanguageTree.hs and LanguageTreeAdventure.hs



Core Data Structures

Command Data Structures

```
data Cmd = Next
         | Back
         | Learn
         | Quiz
         | Progress
         | ExitQuiz
         | Quit
         | Help
         | ChooseLevel   -- New command for choosing a level
         deriving (Show, Eq)
```

Other Data Structures

```
data LangNode = WordNode String String
              | GrammarNode String String
              | QuizNode [QuizQuestion]
              deriving (Show, Eq)

data QuizQuestion = QuizQuestion {
    question :: String,
    answer   :: String
} deriving (Show, Eq)
```

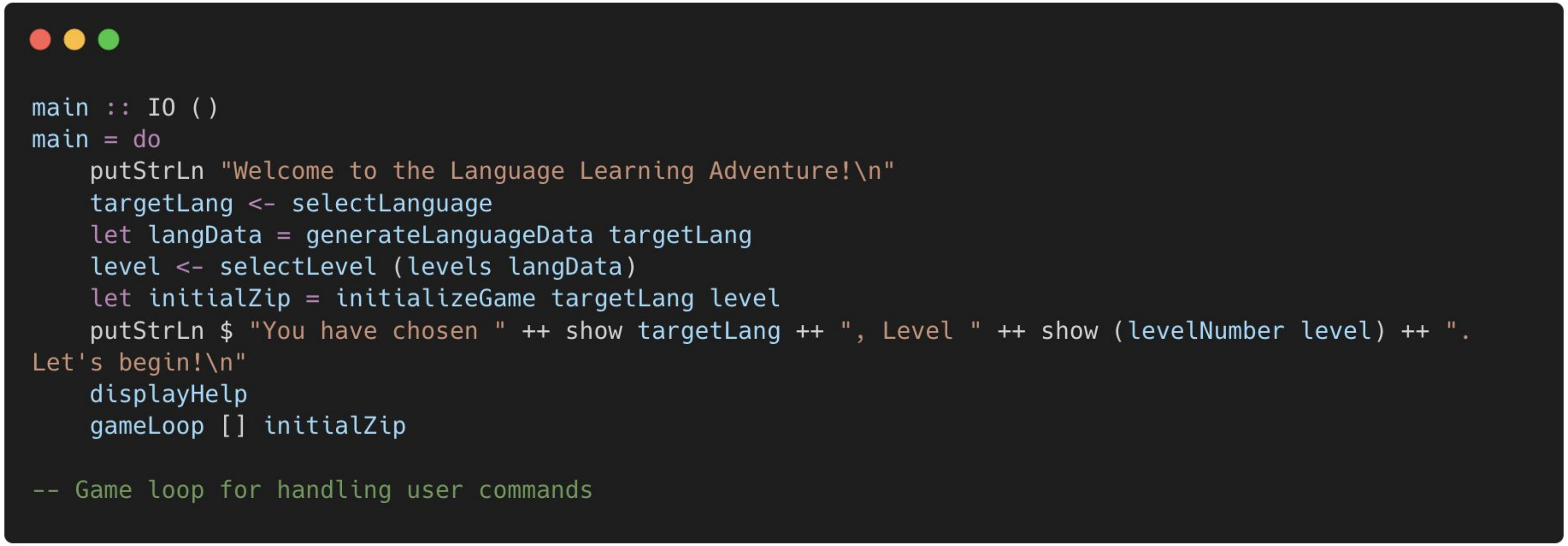
Other Data Structures

```
data LanguageData = LanguageData {
    language :: TargetLanguage,
    levels   :: [Level]
} deriving (Show)

data Level = Level {
    levelNumber :: Int,
    lessons     :: [Lesson]
} deriving (Show)

data Lesson = Lesson {
    title :: String,
    nodes :: [LangNode]
} deriving (Show)
```

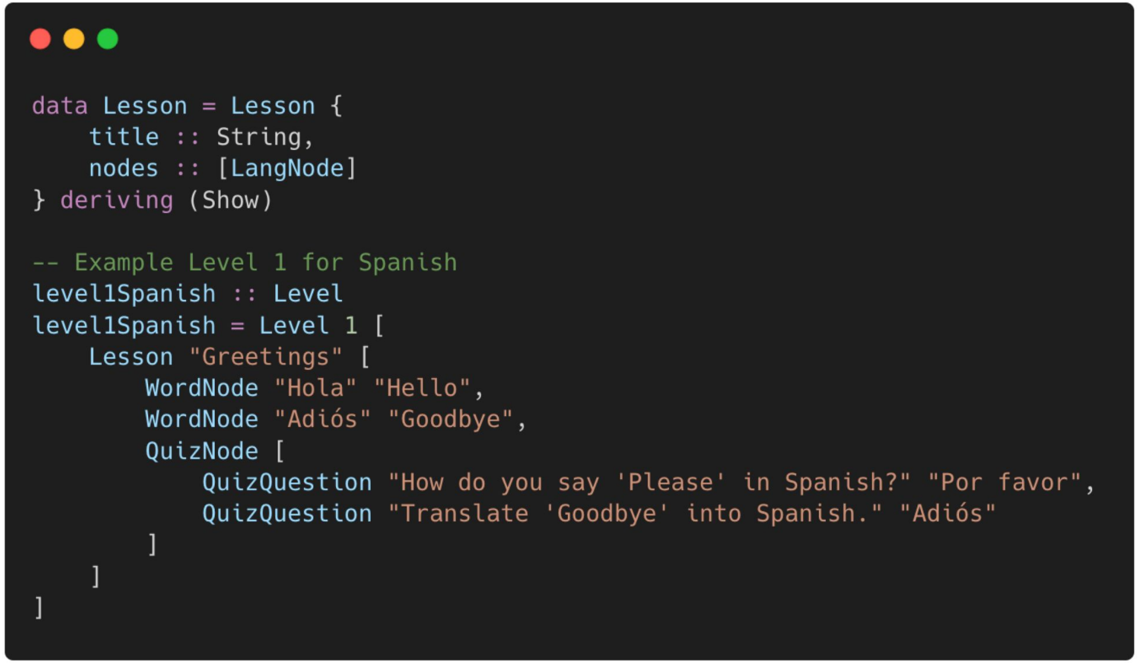
Key Components - Game Loop (Main.hs)



```haskell
main :: IO ()
main = do
    putStrLn "Welcome to the Language Learning Adventure!\n"
    targetLang <- selectLanguage
    let langData = generateLanguageData targetLang
    level <- selectLevel (levels langData)
    let initialZip = initializeGame targetLang level
    putStrLn $ "You have chosen " ++ show targetLang ++ ", Level " ++ show (levelNumber level) ++ ".
Let's begin!\n"
    displayHelp
    gameLoop [] initialZip

-- Game loop for handling user commands
```

Key Components - Language Content (LanguageTree.hs)



```haskell
data Lesson = Lesson {
    title :: String,
    nodes :: [LangNode]
} deriving (Show)

-- Example Level 1 for Spanish
level1Spanish :: Level
level1Spanish = Level 1 [
    Lesson "Greetings" [
        WordNode "Hola" "Hello",
        WordNode "Adiós" "Goodbye",
        QuizNode [
            QuizQuestion "How do you say 'Please' in Spanish?" "Por favor",
            QuizQuestion "Translate 'Goodbye' into Spanish." "Adiós"
        ]
    ]
]
```

Challenges and Solutions

Challenge 1: Implementing Custom Parser Combinators

Challenge 2: Managing Complex State Transitions

Challenge 3: Ensuring Purity and Immutability

```
newtype Parser a = Parser { runParser :: String -> Maybe (a, String) }

instance Functor Parser where
  fmap f (Parser p) = Parser $ \s -> do
    (x, rest) <- p s
    return (f x, rest)
```

```
data LangCxt = InLevel TargetLanguage LevelCxt deriving (Show)

data LevelCxt = LevelCxt {
  currentLevel      :: Level,
  completedLessons  :: [Lesson],
  remainingLessons  :: [Lesson],
  currentLessonCxt  :: Maybe LessonCxt
} deriving (Show)
```

```
data LessonCxt = LessonCxt {
   lessonTitle      :: String,
   completedNodes  :: [LangNode],
   remainingNodes  :: [LangNode]
} deriving (Show)
```

Future Enhancements

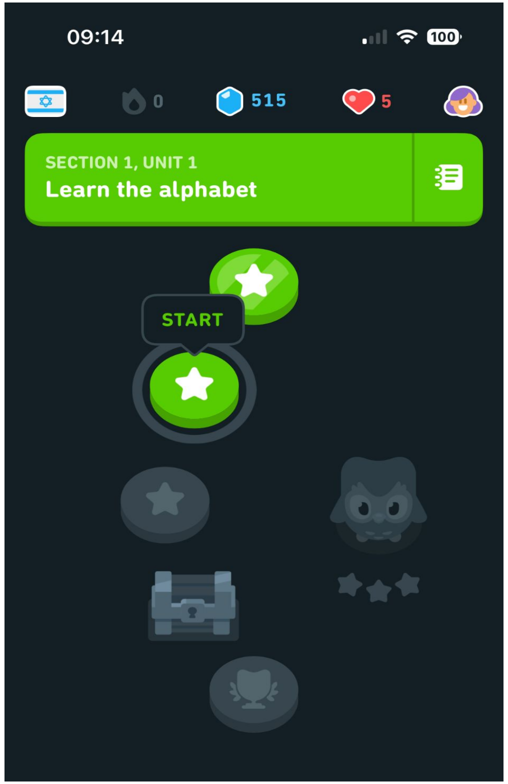
[Multimedia Integration]()

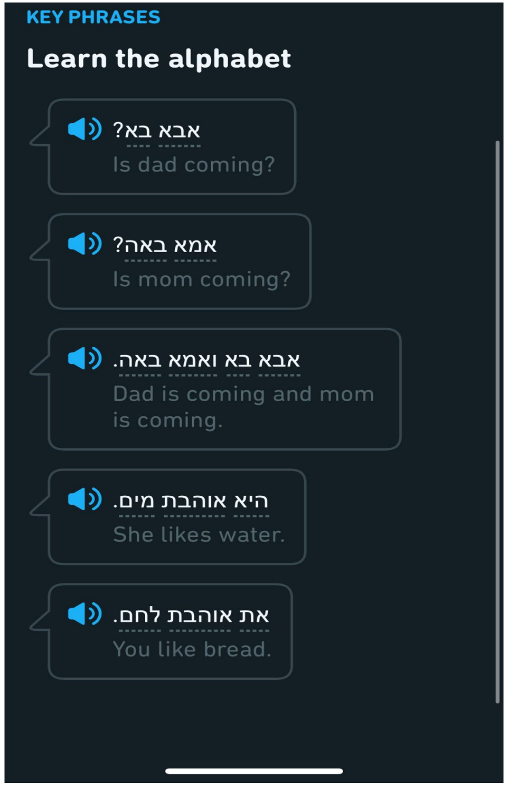
[Spaced Repetition]()

[Personalized Learning Paths]()

[GUI/Web Interface]()

[Additional Languages]()





KEY PHRASES

Learn the alphabet

אבא בא?
Is dad coming?

אמא באה?
Is mom coming?

אבא בא ואמא באה.
Dad is coming and mom is coming.

היא אוהבת מים.
She likes water.

את אוהבת לחם.
You like bread.

DEMO