**De La Salle University**

**Computer Engineering**
**3rd Sem AY 2023**

| **Subject/Course: LBYCPEI: Object Oriented Programming** | | **Professor's Name: Mr. Ruiz** |
|---|---|---|
| **Team Hermes Members:**<br>DIMAPILIS, David Brian S.<br>FERROLINO, Rudd Anthony C.<br>Ong, Aidan Matthew G. | **Section: EQ5** | **Date accomplished: 6/26/2023** |

# TypeCraft: A Groundbreaking Adaptive Typing Program for Enhancing Digital Literacy and Individualized Learning in the Digital Age

## I. Introduction

TypeCraft is a groundbreaking typing program designed to enhance the way students acquire typing skills. Through a blend of gamified learning, adaptive feedback, and tailored lessons, it aims to bolster digital literacy, individualize learning, and integrate seamlessly into existing educational frameworks. Unlike existing applications such as Typing.com, TypingClub, and KeyBlaze, which primarily focus on providing typing lessons and tracking progress, TypeCraft goes a step further. TypeCraft provides a dynamic typing journey, adapting to each learner's skill level and offering personalized feedback to improve accuracy and speed. Its unique feature of customizable lessons allows educators to align typing exercises with curriculum needs, making it a comprehensive tool for boosting computer literacy. This feature sets TypeCraft apart from its competitors, as it empowers educators to tailor the typing program according to their students' needs, enabling seamless integration into existing educational systems. Moreover, TypeCraft's engaging and gamified interface motivates students to continue practicing and progressing. It includes interactive elements, rewards, achievements, and progress tracking, making the learning experience enjoyable and immersive. This level of engagement is a

distinctive aspect of TypeCraft, as it transforms the often monotonous task of learning to type into an exciting and rewarding experience.

Additionally, TypeCraft is designed to optimize the learning process by utilizing efficient techniques that help students learn and improve typing skills in less time. It employs effective exercises, targeted practice, and adaptive learning algorithms to maximize the learning outcome within a shorter duration. This optimization of time is another differentiating factor that gives TypeCraft an edge over similar applications. Suitable for various educational settings, TypeCraft equips students with essential typing skills, paving the way for academic and professional success in the digital age. It's time to unlock the power of typing with TypeCraft.

## II. Methodology

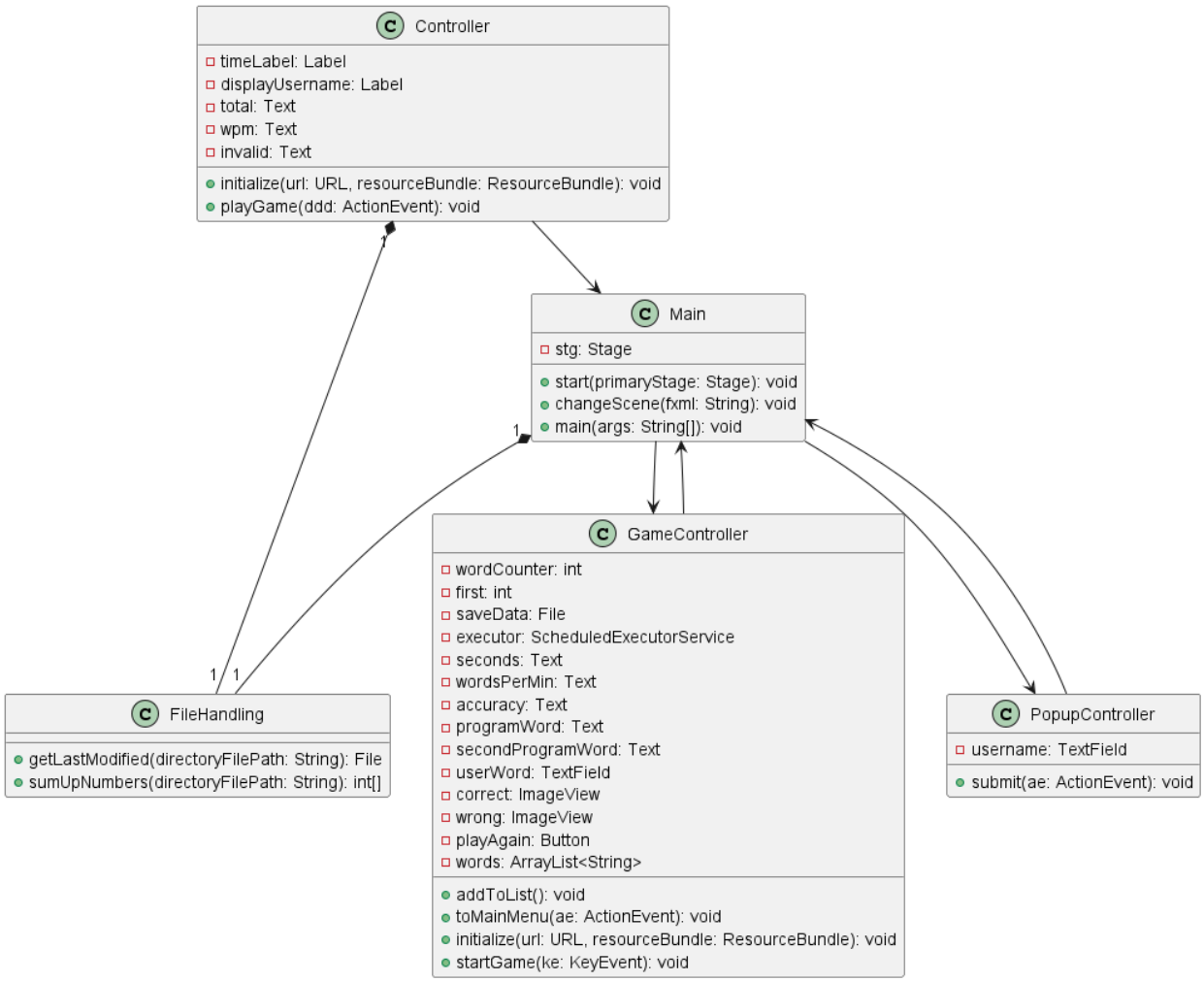
To develop TypeCraft, we aimed to employ a user-centric design approach, focusing on the needs and experiences of the end-users: which is our fellow students and educators. The software will be developed using IntelliJ Java, leveraging its robustness, accessibility, and platform independence. The four pillars of Java - abstraction, encapsulation, inheritance, and polymorphism - will be integral to our development process. Abstraction will be used to hide the complexity of certain operations, encapsulation to protect the data, inheritance to promote code reusability to the mainstream, and polymorphism to allow entities to take on many forms.

The project will be divided into a six-step process:

1. *Requirement Gathering and Analysis:* Understand the needs of the users and define the system requirements.
2. *Design:* Develop the system architecture, including the user interface and database design.
3. *Implementation:* Code the system functionalities using Java.
4. *Testing:* Test the system for bugs and fix them.
5. *Deployment:* Deploy the system for use in educational institutions once approved.
6. *Maintenance:* Provide ongoing support and updates.

## III. Project Description

### Class Diagram:




**IPO: (Input-Process-Output)**

**Input:**

- User login credentials
- User selection from the main menu (start a new lesson, continue a previous lesson, view progress)
- User selection of a specific lesson
- User's typing input during the lesson

**Process:**

- User authentication
- Display of main menu and handling of user selection
- If a new lesson is started, the program displays a list of lessons and starts the selected lesson
- If a previous lesson is continued, the program displays a list of incomplete lessons and resumes the selected lesson from where it was left off
- If progress is viewed, the program generates and displays a progress report

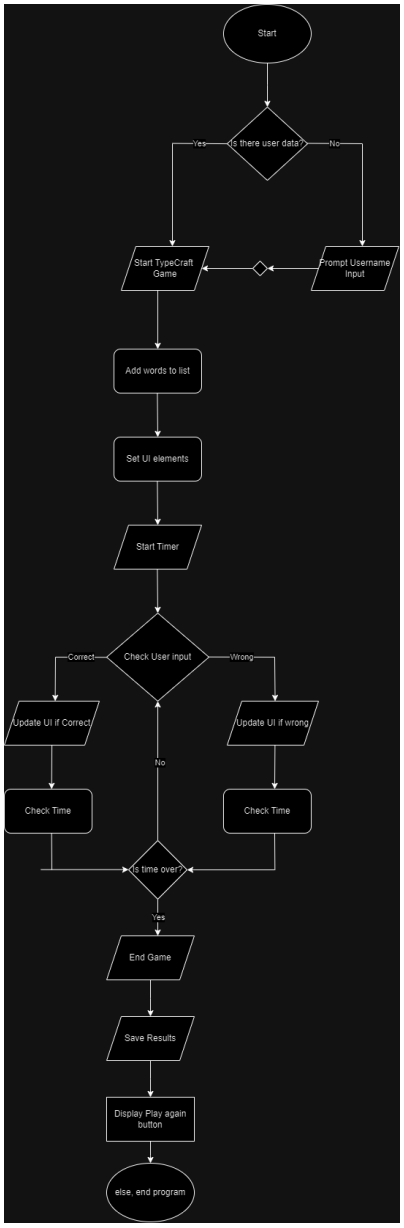
- During a lesson, the program captures the user's typing input, provides real-time feedback, and updates progress upon lesson completion
- The program also manages and displays rewards and achievements based on the user's progress

**Output:**

- Main menu interface
- List of lessons (new or incomplete)
- Typing interface for the selected lesson
- Real-time feedback during the lesson
- Updated progress, rewards, and achievements upon lesson completion
- Progress report when the user chooses to view progress

Description of IPO:
This current IPO is what Team Hermes expects the program to do, it also showcases the major components of the program and how they interact, but the actual implementation would involve more detailed design and coding, which will be done later on.

Flowchart: View FullScreen here




**Description of Flowchart:**

The flowchart for the "TypeCraft" application provides a visual representation of the logical flow and interactions between various components of the code. It consists of different nodes representing methods, decision points, and interactions between classes, connected by arrows

that depict the flow of control. This essay will describe the flowchart in detail, focusing on the main classes: Main, Controller, FileHandling, GameController, and PopupController.

**Main Class**
The flowchart begins with the Main class, where the application is launched. A node represents the start method, initializing the primary stage and setting its properties. An arrow leads to the changeScene method, symbolizing the ability to switch between different scenes.

**Controller Class**
The Controller class is depicted with an initialize node, representing the initialization of the main controller. This node connects to a decision point that checks if user data is available. Depending on the outcome, the flowchart leads to different paths. The playGame method is represented, with two branches leading to different scenes based on whether user data is found.

**FileHandling Class**
The flowchart includes nodes for the FileHandling class, representing methods for file-related operations. The getLastModified method is depicted, followed by the sumUpNumbers method, which reads and processes numbers from files.

**GameController Class**
The GameController class has a complex section in the flowchart. The initialize method sets up the game environment, followed by the addToList method, which reads words from a file. The startGame method is central to this section, with multiple branches representing different game states and outcomes. Decision points are used to handle correct and wrong answers, and various nodes represent the game's timer, feedback animations, and statistics calculations.

**PopupController Class**
The PopupController class is represented with a node for the submit method, handling username input. This part of the flowchart illustrates how the entered username is written to a file and how the scene is changed back to the main menu.

**PUML:**

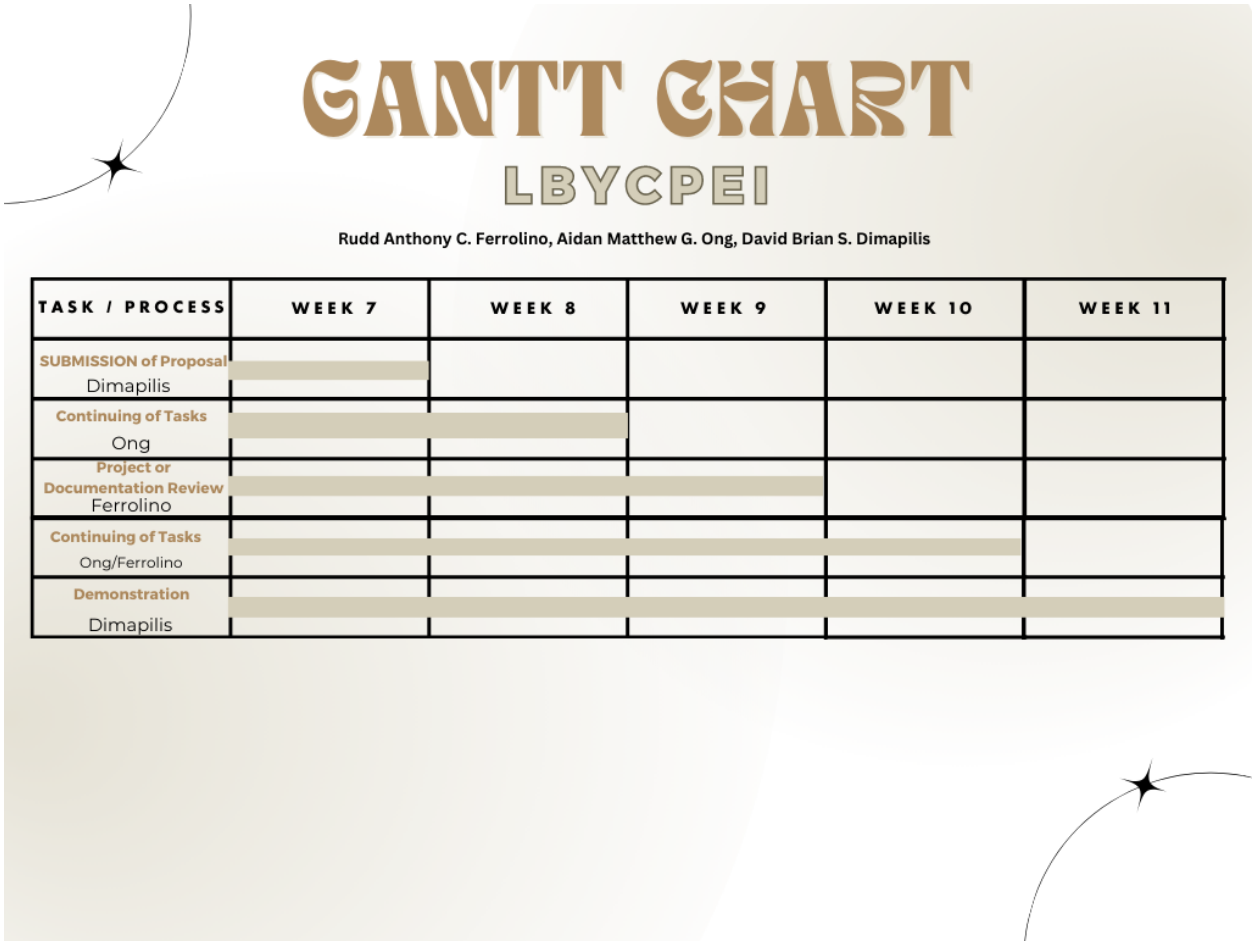
```
@startuml

class Main {
-stg: Stage
+start(primaryStage: Stage): void
+changeScene(fxml: String): void
+main(args: String[]): void
}
```

```
class Controller {
-timeLabel: Label
-displayUsername: Label
-total: Text
-wpm: Text
-invalid: Text
+initialize(url: URL, resourceBundle: ResourceBundle): void
+playGame(ddd: ActionEvent): void
}

class FileHandling {
+getLastModified(directoryFilePath: String): File
+sumUpNumbers(directoryFilePath: String): int[]
}

class GameController {
-wordCounter: int
-first: int
-saveData: File
-executor: ScheduledExecutorService
-seconds: Text
-wordsPerMin: Text
-accuracy: Text
-programWord: Text
-secondProgramWord: Text
-userWord: TextField
-correct: ImageView
-wrong: ImageView
-playAgain: Button
-words: ArrayList<String>
+addToList(): void
+toMainMenu(ae: ActionEvent): void
+initialize(url: URL, resourceBundle: ResourceBundle): void
+startGame(ke: KeyEvent): void
}

class PopupController {
-username: TextField
+submit(ae: ActionEvent): void
}

Main --> GameController
Main --> PopupController
Main "1" *-- "1" FileHandling
Controller --> Main
Controller "1" *-- "1" FileHandling
GameController --> Main
```

```
PopupController --> Main

@enduml
```

IV. Deliverables

GANTT CHART:




V. Evaluation

*User Interface and Experience:* The design of the interface should be intuitive and easy to navigate, even for users with little to no experience with similar software. It should also be engaging to maintain user interest.

*Adaptability and Customizability:* The program should have the ability to adapt to a user's skill level and provide customizable lessons. It should be evaluated on how well it caters to different user needs and learning styles.

*Learning Effectiveness:* This is one of the most important criteria. It measures how effective the typing program is in improving the typing skills of the students. Metrics like improvement in typing speed and accuracy over time can be used.

*Feedback and Performance Tracking:* The program should provide clear, specific, and constructive feedback on the user's performance. It should also track user performance over time to show progress.

*Integration with Existing Systems*: If the typing program is being used in an educational institution, it needs to be compatible with the existing systems in the institution.

*Reliability and Stability:* Since it's a local program, the stability of the software is important. It should not crash frequently, and should be able to recover gracefully in case of any failures.

## VI. Conclusion

The TypeCraft project is a significant initiative to enhance digital literacy by creating an engaging and adaptive educational typing program. Using gamification and personalized feedback, it facilitates individualized learning and fosters typing mastery, filling a gap in many educational institutions that lack an engaging and effective platform for teaching these essential skills. Notably, its local, non-online nature broadens its accessibility, making it an invaluable tool for schools or individuals with limited internet access. Overall, TypeCraft offers a comprehensive solution for computer literacy that supports academic success and prepares students for future professional endeavors in a digital-centric world.

VII. References

*Typing Game for Java*. GitHub. (n.d.). https://github.com/topics/typing-game?l=java

Codementor. (n.d.). *Speed typing game*. DevProjects. https://www.codementor.io/projects/web/speed-typing-game-c51led1afn

GeeksforGeeks. (2022, August 1). *Design a typing speed test game using JavaScript*. GeeksforGeeks. https://www.geeksforgeeks.org/design-a-typing-speed-test-game-using-javascript/