Bug Bytes

Polus Banyameen, Maria Leon, Sydney Lum
Computer Science and Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
pbanyameeen@oakland.edu, mleoncarrera@oakland.edu, slum@oakland.edu

Bug Bytes is a website that assists users with their first steps into the world of programming. It provides users with a guided lesson and questions that will help them work on their first programming fundamentals, such as variables and writing basic functions. As the world of programming continues to grow, diving into coding can feel overwhelming for beginners. Our goal is to be the helping hand that empowers individuals to begin developing their skills.

I. Introduction

Bug Bytes is a website that offers users interactive basic lessons in programming with Java, which could be their first coding language. The site will guide each user into a lesson on how to install an IDE, explain what an IDE is, and how to share their final projects. Apart from logistics, Bug Bytes will teach users coding using proper syntax, utilizing variables and functions. They will learn to write the famous "Hello World!" statement and explore essential functions like main, print, and more! To make the experience more entertaining, hand-drawn ape companions will accompany the user along with aesthetic backgrounds. Bug Bytes is built to be beginner-friendly and easy to use, making it perfect for first time users.

Nowadays, technology has been advancing at an exponential rate and the demand for jobs requiring basic programming knowledge continues to grow. For some, programming comes naturally- but for many, it can be a challenging journey, especially when learning independently [1]. Beginners can abandon a task when hitting obstacles, which is why we created Bug Bytes! Our goal is to make learning programming easier and enjoyable, especially for those who are starting out and may feel overwhelmed. We want to give beginners, who are struggling with programming, a way to learn effortlessly and have fun. The idea for Bug Bytes was inspired by our similar experiences. At some point, each of us has struggled while learning the programming languages we know today, and we wanted to build a platform that could ease the learning process for others. We were inspired by the many ways people learn linguistic languages, and we figured, why not apply a similar approach to programming languages instead? With Bug Bytes, more people can begin their coding journey with confidence. Those who already have some experience will find it useful, whether they are brushing up on their own skills or helping others take their first steps into coding.

II. Project Details

A. Project diagram

The foundational architecture of our entire project follows a straightforward process, shown in **Figure 1**. When a user provides input- such as uploading code to answer any of the lesson questions- that input is sent to the backend for processing. After the backend receives it, it follows one or both of the following operations (sometimes in sequence, other times, simultaneously). First, it runs its internal logic to determine the correct response based on the input. Second, the backend queries the MySQL database- for instance, to verify whether the submitted code fulfilled all specific criteria. The backend may query the database before, during, or after processing the input, depending on the nature of the request. Once all necessary operations are completed, the backend sends a response to the frontend. The frontend then dynamically presents feedback to the user and explains what happened and how their input was handled. This interaction pattern repeats for any user input that requires backend processing. **Figure 2** provides a more detailed example.In this case, a user is trying to register with the website. In this scenario, a user attempts to register on the website. If they initially submit information (such as a username or email) that already exists in the database, the backend returns an error. After providing a unique username and email, the backend and database validate the input, and the user is redirected to the homepage with access to features such as the lesson and shop pages.

A. Project requirements

Requirement 1. A website structured around a guided lesson plan, with integrated questions to assess user comprehension. Each lesson is divided into multiple sections, and after each section, users are tested with either fill-in-the-blank or multiple-choice questions. See **Figures 3** and **4** for examples of a typical "learning" section and its corresponding question segment. When a user submits an answer, a request is sent to the backend for processing, and the result is reflected on the frontend, for the user. In addition to these questions, each lesson concludes with a code-writing exercise designed to reinforce the concepts covered. These exercises come with specific requirements and objectives—see **Figure 5** for a sample. We have implemented login and registration pages, allowing users to create accounts and track their progress. Once logged in, users can resume from where they left off. See **Figures 6** and **7** for interface examples. While it's not an industry standard, we opted to use local storage to temporarily store user progress and certain pieces of information. This decision was made to streamline development within our current skill set, as implementing cookies would have significantly delayed progress under our time constraints. See **Figure 8** for an illustration. However, we still maintain persistent data on the backend to ensure progress is not lost when the user logs out.

Requirement 2. The backend is responsible for handling key functionalities such as question validation, account management, and user progression tracking. It always remains on standby to receive requests from the frontend and respond accordingly. This includes saving data in the MySQL database or processing logic that the frontend cannot handle alone. As shown in **Figure 9**, a code segment handles an API call that takes four parameters: the user's selected answer, the question ID, the user ID, and the current list of answered questions. The backend processes this input and interacts with the MySQL database as needed. If the user selects the correct answer, the backend adds the corresponding question ID to the user's list of completed questions. It then returns a response containing the updated progression and a message confirming the correct answer. If the answer is incorrect, the backend returns the current progression (unchanged) and a message indicating the answer was wrong. Once the user selects the correct answer, the backend updates the MySQL database by adding a new row into the user_completed_questions table. This row includes the user ID—linked to the users table—and the ID of the correctly answered question. See **Figures 10 & 11** for an example of this interaction.

Requirement 3. A key feature of our platform is the store system, which ties directly into user progression and includes its own in-app economy. Users earn virtual currency by answering questions correctly. This currency can then be used to purchase cosmetic items that customize their ape companion. When a purchase is made, the user's coin balance is reduced accordingly. The ape's appearance dynamically updates on-screen based on the items selected. As shown in **Figure 12**, users can preview how an item will look on their ape companion before purchasing it. Once an item is bought, the backend updates the user's list of owned items and deducts the appropriate amount of currency. The frontend reflects these changes in real time. **Figures 13** and **14** demonstrate how this interaction is processed and recorded in both the frontend and the database. In addition, users can personalize their ape companion's appearance at any time. These customization preferences are stored per user, ensuring that updates, such as removing a hat, are preserved even after the user logs out. When they log back in, their selected appearance is restored automatically. **Figures 15** and **16** illustrate this by showing how user preferences are saved and reflected in the database.

B. Frontend Design

Bug Bytes is designed to be a user-friendly website that provides easy and accessible learning across all devices. We developed the site using HTML, CSS, and JavaScript, focusing on creating a clean, intuitive, and engaging interface. The website structure is built with semantic HTML5 elements, allowing for better organization and enhanced support for graphics and visual effects to enrich the user's learning experience. CSS is used to design consistent and visually appealing layouts, while JavaScript combines interactive features that enhance the site's usability. JavaScript enables communication with the backend, developed in Spring Boot, by sending and receiving requests based on user interactions. For the visual design, we incorporated illustrations rendered in Procreate, adding a personal and creative touch to the site. We used Figma to design custom UI elements such as buttons [2][3]. For guidance of some aspects of the frontend layout, we referenced other external resources. We used FreeCSS.com for inspiration on HTML and CSS implementations, especially given that these were areas where we had less experience [4]. Specifically, we referenced the Namari template by user ShapingRain to learn how to implement a fixed top navigation bar [5]. Additionally, we utilized the LifeStyle template by Bootstrap51 as a reference for structuring our login and registration pages [6].

C. Backend Design

The backend of Bug Bytes was developed using Java as the primary programming language, with Spring Boot serving as the framework. Spring Boot is one of the most widely used Java frameworks for building web applications, offering a fast, flexible, and productive development environment. It simplifies the creation of stand-alone, production-grade applications through its powerful set of extensions and libraries [7]. To streamline feature implementation, we integrated several key Spring Boot dependencies, including the MySQL Connector, Spring Boot DevTools, and Spring Boot Security. These tools enable us to build RESTful APIs that communicate with the frontend via JSON, ensuring a dynamic and responsive user experience [8]. Functionalities such as account creation, answer validation, and code compilation are handled securely through the backend. Offloading these operations from the frontend ensures data integrity and prevents misuse or manipulation. Relying solely on the frontend for these tasks would not only limit functionality but also create security vulnerabilities. For persistent data storage, we selected MySQL, a widely supported relational database system compatible with Spring Boot. MySQL allows us to reliably store and retrieve user data—such as account information, progress, and completed lessons—so users can seamlessly continue where they left off. Although maintaining a persistent server is necessary for a production environment, our website is currently designed to run locally. This approach allows us to focus on learning and building core features, as this is our first project of this scope and complexity. The server is managed using Apache Maven, which handles dependency management and project builds.

III. Group Responsibilities

Polus developed multiple question frameworks for the website, including multiple-choice, fill-in-the-blank, and short-code writing questions. He also implemented a file-checking system that runs submitted code and verifies its correct answer before awarding the user a correct mark. Additionally, Polus completed the frontend for the shop page, adding three hats for users to purchase. He also worked on creating lesson plans and used CSS and JavaScript to ensure the pages were visually appealing and functional. Moreover, he contributed to the development of the login and registration pages, working on both the frontend and backend.

Sydney designed and implemented the database using MySQL to securely store and encrypt user identification, usernames, passwords, and progression data. She optimized the connectivity between Java and MySQL [12][13]. Sydney arranged the layout for the site's pages: the homepage, login page, lesson page, and shop page. In addition, she illustrated three different ape companions and an amusing homepage background. She created the "About Me" page [25] and added a dropdown menu [29] to the main navigation using CSS, HTML, and JavaScript, ensuring compatibility with Java.

Maria created the lesson plan structure and contributed common menu headers. She worked with CSS, HTML, and JavaScript to establish a uniform and presentable style across the site. Maria designed beginner-level lesson plans, contributed to some functionalities on the shop page, and worked on the development of the homepage frontend and its utilities.

IV. Technologies Used

For this project, we used multiple programs to make this website functional:

Front end:

- JavaScript
  - Prism Library for code coloring
- HTML/CSS
- Procreate

Back end:

- IntelliJ
- Visual Studio Code
- MySQL
- Java
  - SpringBoot
  - Apache Maven
- REST API
- Terminal

Our primary form of communication was through GroupMe [14], where we could efficiently share information among team members, allowing for short, quick responses to questions and updates. For more in-depth discussions, we use Google Meet [15], which enabled us to hold online meetings without the need to meet in-person. Given our busy and conflicting schedules, the ability to quickly meet online has been invaluable. We met at least once or twice a week to stay updated on each team member's progress towards the website.

For task organization and responsibility assignment, we relied on Google Sheets [16]. This helped us track our progress, keeping everyone informed about ongoing tasks, completed work, and updates. We also used a shared Google Drive for non-code documents, such as milestone sheets and planning materials.

For code management, we utilized a shared GitHub repository [17]. GitHub allows us to upload individual code, securely merge it, and compare changes. Each team member worked on their own branch, testing new features before merging them into the main branch. Sydney used Git and Terminal to stage, upload, and commit code to GitHub.

V. Next Steps

If given more time, we would expand the lesson plans to cover more advanced Java concepts beyond the basics. These lessons would be organized into different levels of understanding: beginner, intermediate, advanced, expert, and master levels, each with new content corresponding to the user's progression. We would introduce more interactive functionalities and segments to the website to further engage users and increase session duration. Although we initially hesitated to include a shop feature, we were able to implement a basic version due to time constraints. With more time, we could enhance the shop by adding more items and accessories for the apes, as well as categorizing them for easier navigation. Additionally, we envision adding a navigation or reference bar, where users could search for unfamiliar terms and directly reference the lessons in which those terms are used. This would help users gain a deeper understanding of the concepts.

VI. Ethical Impact

Bug Bytes is a website designed to help individuals who are new to coding, providing a comprehensive introduction to the basics of Java. Our platform serves as a starting point for those struggling to find a way to learn. Through Bug Bytes, users can access informative resources and practice foundational Java skills in an interactive and fun environment.

As we offer account creation, Bug Bytes handles sensitive user information, including email addresses, usernames, and passwords. To ensure the security of this data, we used Bcrypt encryption for passwords, providing an added layer of protection against potential data breaches and safeguarding user privacy.

As an educational platform, we have a responsibility to ensure the accuracy and currency of the information provided. Inaccurate or outdated content could mislead users, create unnecessary learning challenges, and undermine their coding projects. It could also damage our credibility as a trusted resource. By maintaining high standards of accuracy, we can offer genuine support and help users succeed in their learning journey.

VII. Generative AI Usage (optional)

We used ChatGPT to better understand the mechanics of our program, particularly in backend design. We used AI to gain insights into API usage, including which APIs would be the best for our needs. ChatGPT also helped us familiarize ourselves with frameworks like Spring Boot for the backend and JavaScript for the frontend, ensuring we were well-prepared to implement the project effectively [18].

In addition to backend and frontend frameworks, we used ChatGPT to guide us through the process of securely decrypting passwords, which ultimately led us to the BCrypt library in Java for encryption [19]. Furthermore, ChatGPT assisted us in developing the lesson plans for our course, helping us create up-to-date, beginner-friendly content to support our users' learning experience [20][21].

VIII. References

[1] https://www.weforum.org/stories/2025/01/future-of-jobs-report-2025-the-fastest-growing-and-declining-jobs/
[2] https://www.figma.com/
[3] https://procreate.com/
[4] https://www.free-css.com/
[5] https://www.free-css.com/free-css-templates/page294/namari
[6] https://www.free-css.com/free-css-templates/page201/lifestyle
[7] https://rollbar.com/blog/most-popular-java-web-frameworks/
[8] https://www.youtube.com/watch?v=9SGDpanrc8U
[9] https://stackoverflow.com/questions/34847231/how-to-save-progress-in-an-html-game
[10] https://stackoverflow.com/questions/6972092/ios-how-to-store-username-password-within-an-app
[11] https://xdaforums.com/t/how-to-save-users-data-individually-and-provide-them-if-someone-requests-it.3810674/
[12] https://www.geeksforgeeks.org/java-database-connectivity-with-mysql/
[13] https://stackoverflow.com/questions/2839321/connect-java-to-a-mysql-database
[14] https://groupme.com/
[15] https://meet.google.com/
[16] https://workspace.google.com/products/sheets/
[17] https://github.com/
[18] https://chatgpt.com/share/67b4b2cb-a5c4-800b-b090-2a92e8544e63
[19] https://chatgpt.com/share/67d9a1d6-f5e8-800b-8a6f-b91f4f8f5148
[20] https://chatgpt.com/share/67f94c54-b21c-800b-a73d-d42d98ca4fb8
[21] https://chatgpt.com/share/67ff0ac3-1e10-800b-9b6a-e234797d2a59
[22] https://www.w3schools.com/css/default.asp
[23] https://www.w3schools.com/js/default.asp
[24] https://www.w3schools.com/html/default.asp
[25] https://www.geeksforgeeks.org/design-a-web-page-using-html-and-css/#
[26] https://prismjs.com/
[27] https://www.youtube.com/watch?v=MDjOH8fw-nw
[28] https://www.youtube.com/watch?v=S-VeYcOCFZw&list=PLqLGjynmyqwWOyrhl9BSKZSjpEluUzT0c&index=22

Appendix

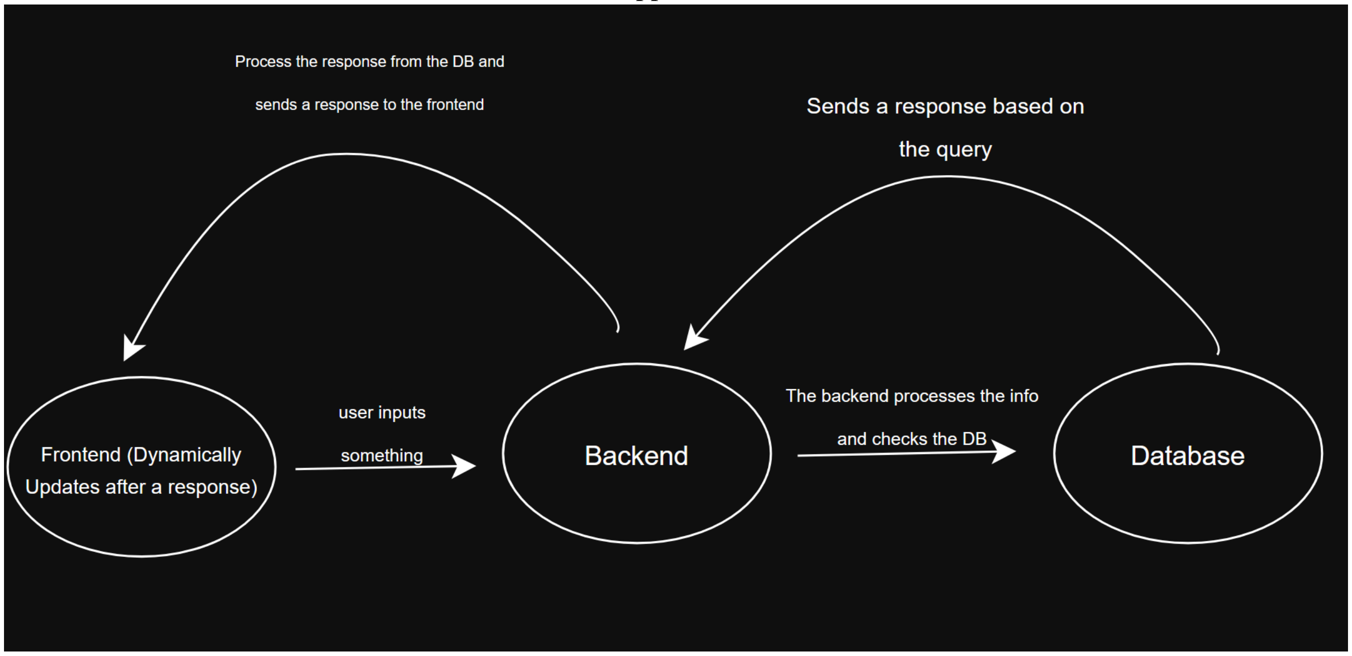

Process the response from the DB and sends a response to the frontend
Sends a response based on the query
user inputs something
The backend processes the info and checks the DB
Frontend (Dynamically Updates after a response)
Backend
Database

Figure 1: Architecture of the Project

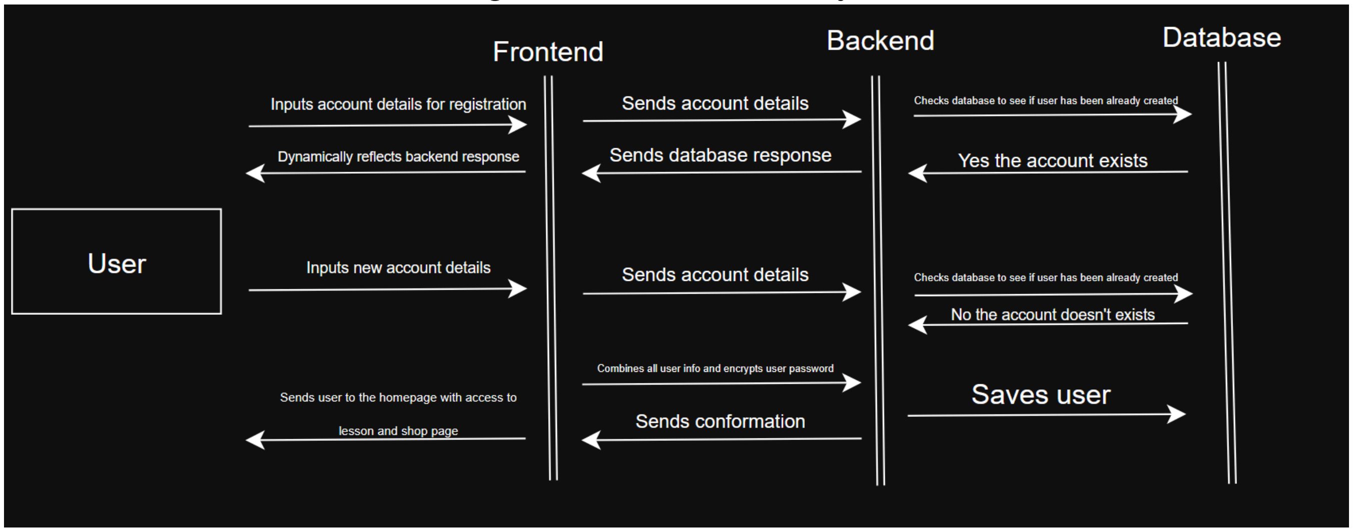
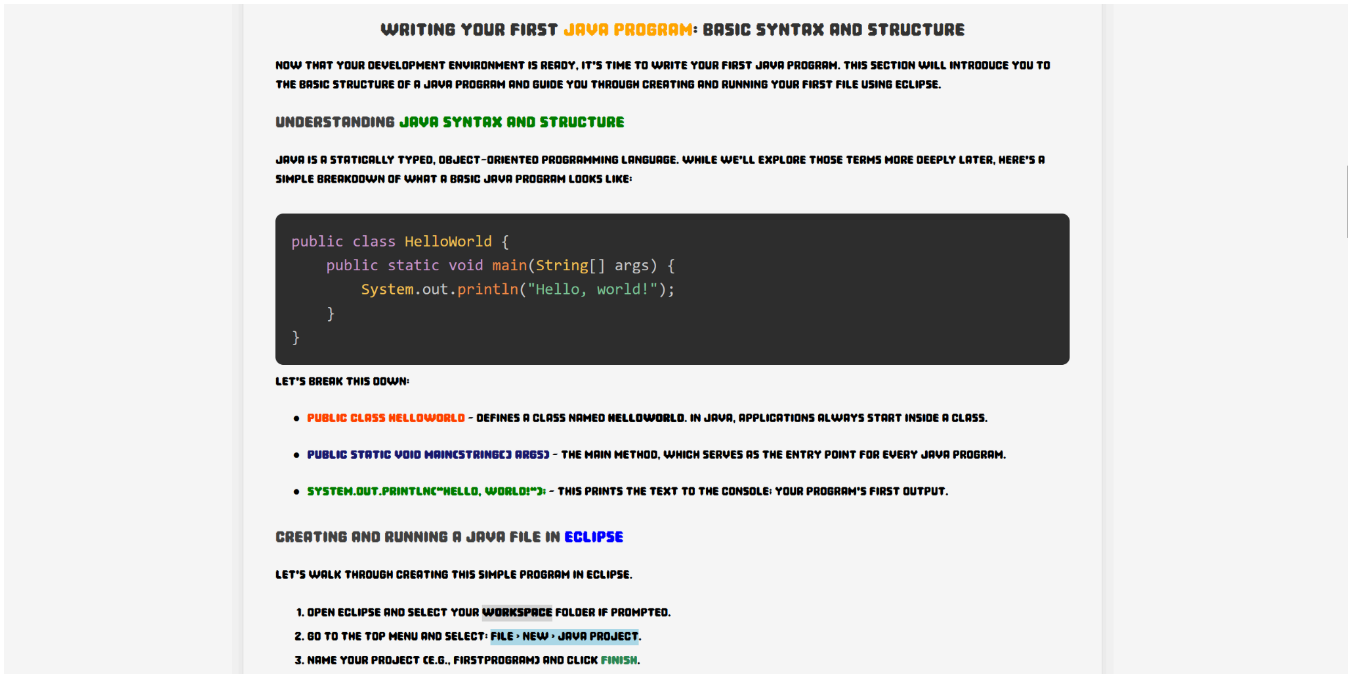

Frontend
Backend
Database
User
Inputs account details for registration
Sends account details
Checks database to see if user has been already created
Dynamically reflects backend response
Sends database response
Yes the account exists
Inputs new account details
Sends account details
Checks database to see if user has been already created
No the account doesn't exists
Combines all user info and encrypts user password
Saves user
Sends user to the homepage with access to lesson and shop page
Sends conformation

Figure 2: A user trying to register


WRITING YOUR FIRST JAVA PROGRAM: BASIC SYNTAX AND STRUCTURE

NOW THAT YOUR DEVELOPMENT ENVIRONMENT IS READY, IT'S TIME TO WRITE YOUR FIRST JAVA PROGRAM. THIS SECTION WILL INTRODUCE YOU TO THE BASIC STRUCTURE OF A JAVA PROGRAM AND GUIDE YOU THROUGH CREATING AND RUNNING YOUR FIRST FILE USING ECLIPSE.

UNDERSTANDING JAVA SYNTAX AND STRUCTURE

JAVA IS A STATICALLY TYPED, OBJECT-ORIENTED PROGRAMMING LANGUAGE. WHILE WE'LL EXPLORE THOSE TERMS MORE DEEPLY LATER, HERE'S A SIMPLE BREAKDOWN OF WHAT A BASIC JAVA PROGRAM LOOKS LIKE:

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

LET'S BREAK THIS DOWN:

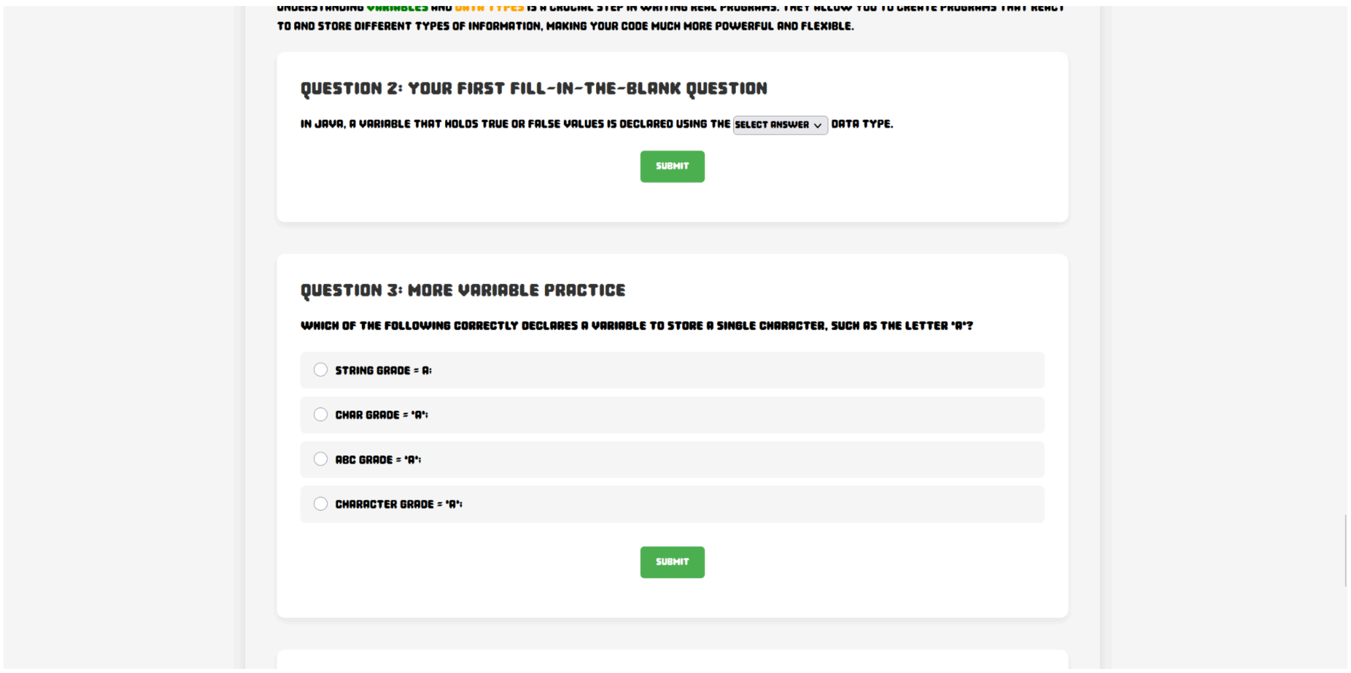
- PUBLIC CLASS HELLOWORLD – DEFINES A CLASS NAMED HELLOWORLD. IN JAVA, APPLICATIONS ALWAYS START INSIDE A CLASS.
- PUBLIC STATIC VOID MAIN(STRING[] ARGS) – THE MAIN METHOD, WHICH SERVES AS THE ENTRY POINT FOR EVERY JAVA PROGRAM.
- SYSTEM.OUT.PRINTLN("HELLO, WORLD!"); – THIS PRINTS THE TEXT TO THE CONSOLE: YOUR PROGRAM'S FIRST OUTPUT.

CREATING AND RUNNING A JAVA FILE IN ECLIPSE

LET'S WALK THROUGH CREATING THIS SIMPLE PROGRAM IN ECLIPSE.

1. OPEN ECLIPSE AND SELECT YOUR WORKSPACE FOLDER IF PROMPTED.
2. GO TO THE TOP MENU AND SELECT: FILE › NEW › JAVA PROJECT.
3. NAME YOUR PROJECT (E.G., FIRSTPROGRAM) AND CLICK FINISH.

Figure 3: "Writing Your First Java Program" lesson section of the first lesson.


UNDERSTANDING VARIABLES AND DATA TYPES IS A CRUCIAL STEP IN WRITING REAL PROGRAMS. THEY ALLOW YOU TO CREATE PROGRAMS THAT REACT TO AND STORE DIFFERENT TYPES OF INFORMATION, MAKING YOUR CODE MUCH MORE POWERFUL AND FLEXIBLE.

QUESTION 2: YOUR FIRST FILL-IN-THE-BLANK QUESTION

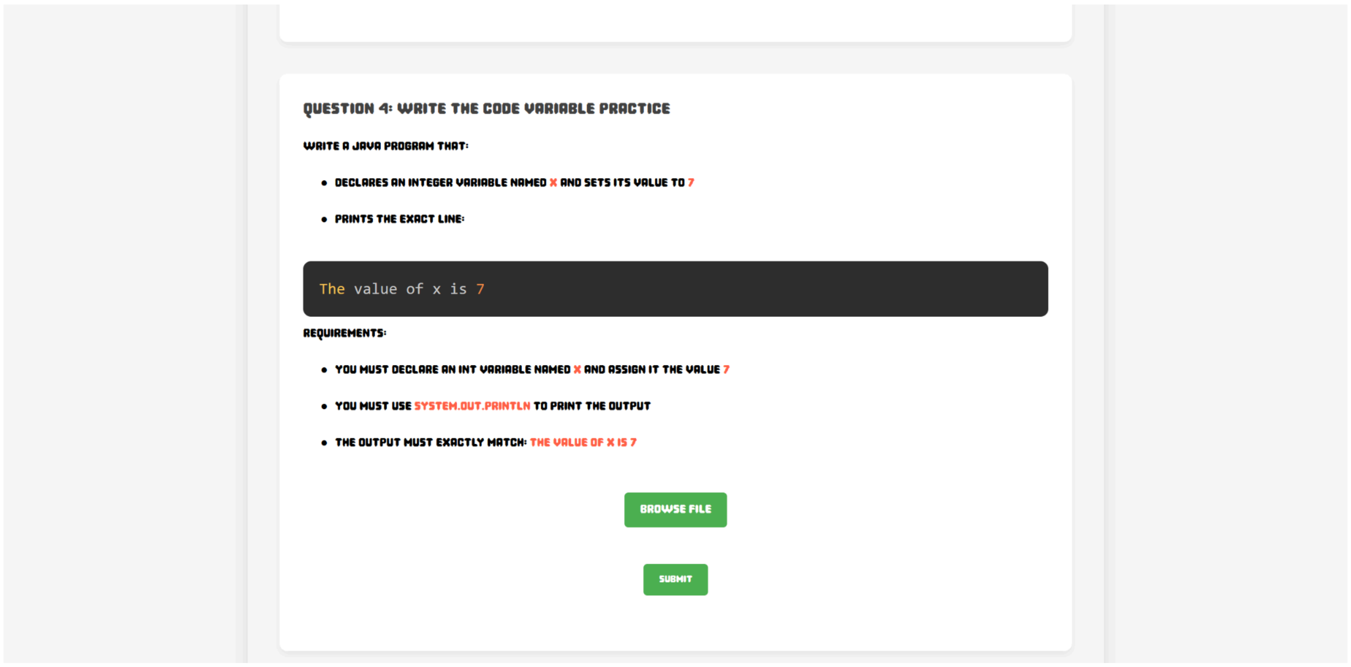
IN JAVA, A VARIABLE THAT HOLDS TRUE OR FALSE VALUES IS DECLARED USING THE SELECT ANSWER DATA TYPE.

QUESTION 3: MORE VARIABLE PRACTICE

WHICH OF THE FOLLOWING CORRECTLY DECLARES A VARIABLE TO STORE A SINGLE CHARACTER, SUCH AS THE LETTER 'A'?

- STRING GRADE = A;
- CHAR GRADE = 'A';
- ABC GRADE = 'A';
- CHARACTER GRADE = 'A';

Figure 4: A fill-in-the-blank and multiple-choice question following the Variables and Data Types section of lesson one.



QUESTION 4: WRITE THE CODE VARIABLE PRACTICE

WRITE A JAVA PROGRAM THAT:

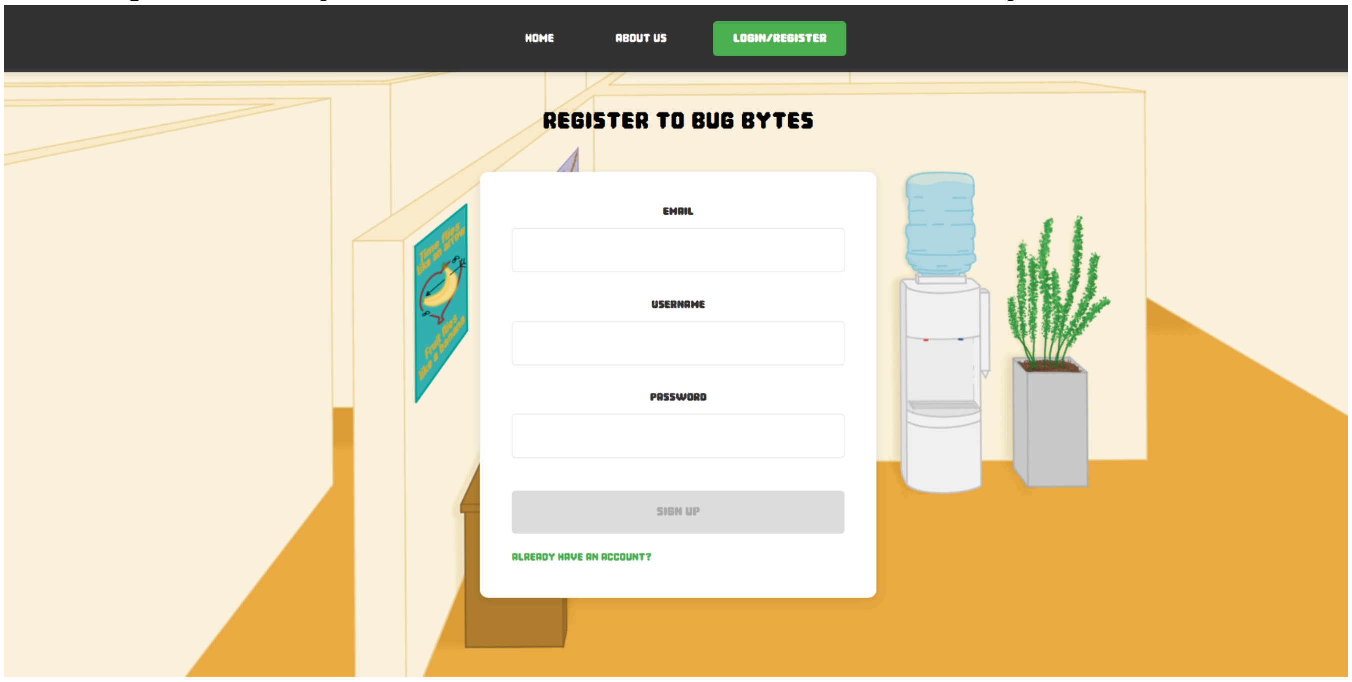
- DECLARES AN INTEGER VARIABLE NAMED X AND SETS ITS VALUE TO 7
- PRINTS THE EXACT LINE:

```
The value of x is 7
```

REQUIREMENTS:

- YOU MUST DECLARE AN INT VARIABLE NAMED X AND ASSIGN IT THE VALUE 7
- YOU MUST USE SYSTEM.OUT.PRINTLN TO PRINT THE OUTPUT
- THE OUTPUT MUST EXACTLY MATCH: THE VALUE OF X IS 7

Figure 5: The last question of lesson one asks the user to declare a variable and print out its content.



REGISTER TO BUG BYTES

EMAIL

USERNAME

PASSWORD

Figure 6: The register screen. Appears after the user selects the green Login/Register button on the navigation bar.

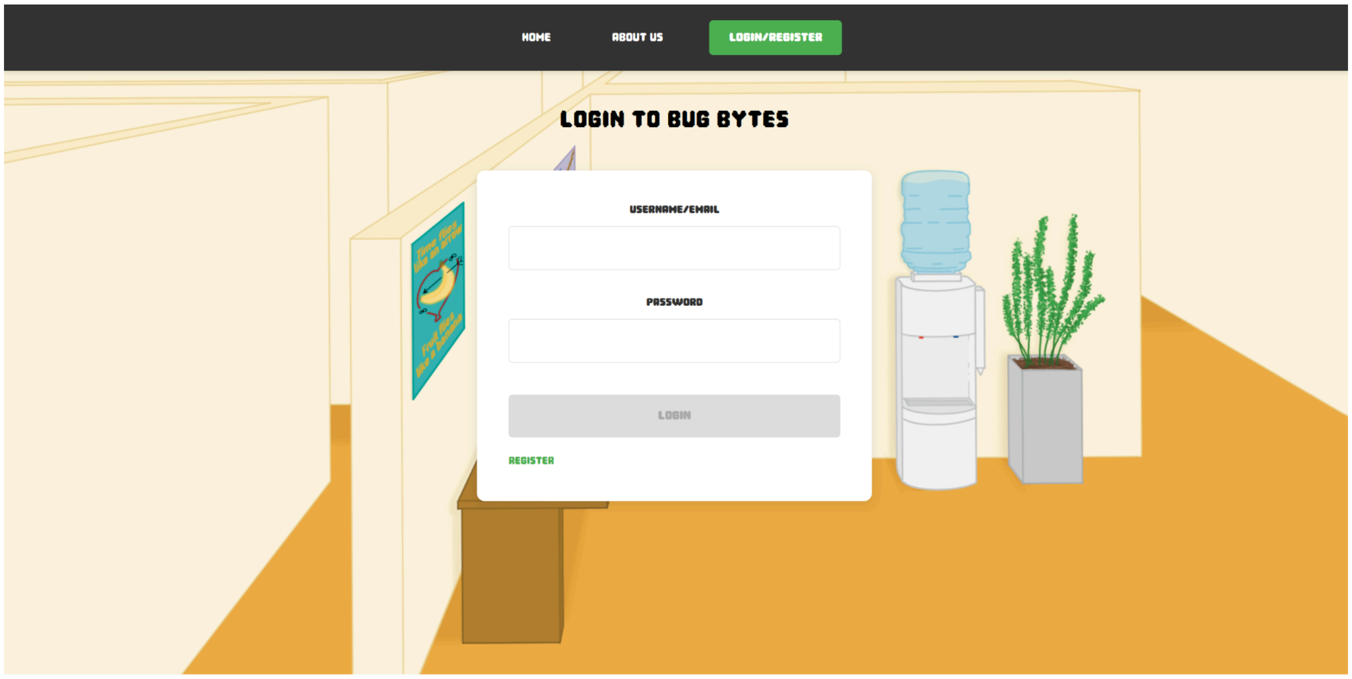
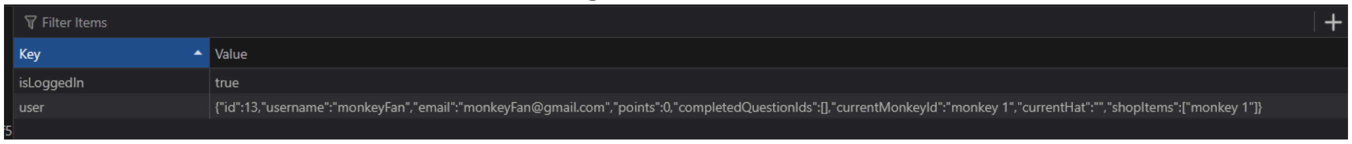
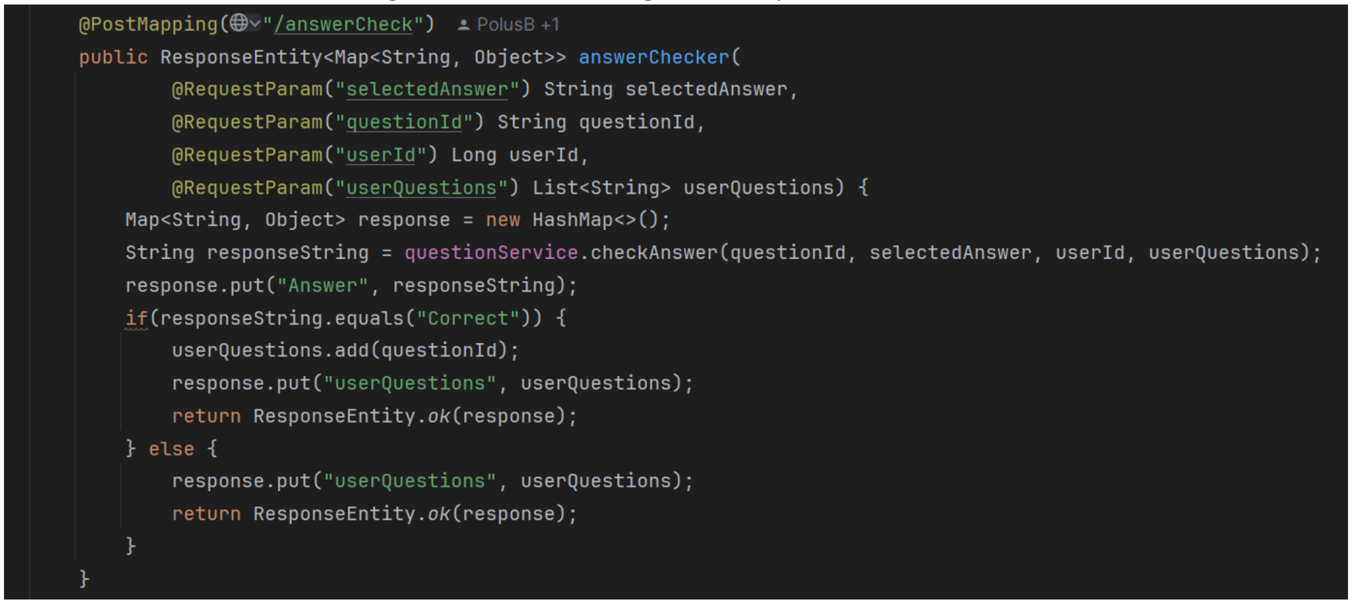


Figure 7: The register screen. Appears after the user selects the green "Already have an account?" hyperlink on the registration screen.



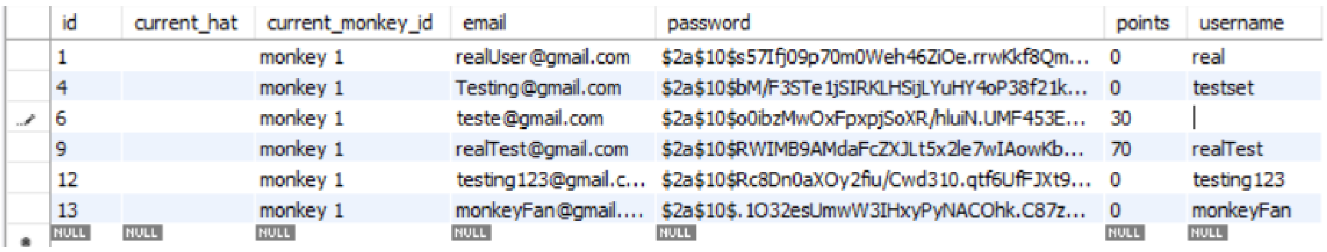
Key	Value
isLoggedIn	true
user	{"id":13,"username":"monkeyFan","email":"monkeyFan@gmail.com","points":0,"completedQuestionIds":[],"currentMonkeyId":"monkey 1","currentHat":"","shopItems":["monkey 1"]}

Figure 8: The local storage of a newly created account.



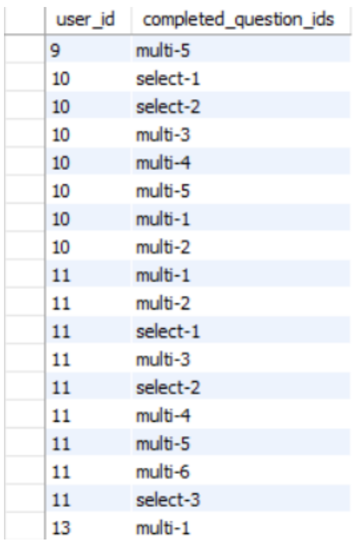
```
@PostMapping("/answerCheck")  PolusB +1
public ResponseEntity<Map<String, Object>> answerChecker(
        @RequestParam("selectedAnswer") String selectedAnswer,
        @RequestParam("questionId") String questionId,
        @RequestParam("userId") Long userId,
        @RequestParam("userQuestions") List<String> userQuestions) {
    Map<String, Object> response = new HashMap<>();
    String responseString = questionService.checkAnswer(questionId, selectedAnswer, userId, userQuestions);
    response.put("Answer", responseString);
    if(responseString.equals("Correct")) {
        userQuestions.add(questionId);
        response.put("userQuestions", userQuestions);
        return ResponseEntity.ok(response);
    } else {
        response.put("userQuestions", userQuestions);
        return ResponseEntity.ok(response);
    }
}
```

Figure 9: The function used for answer checking both fill-in-the-blank and multiple-choice questions. Uses a Post REST API call.



id	current_hat	current_monkey_id	email	password	points	username
1		monkey 1	realUser@gmail.com	$2a$10$s57Ifj09p70m0Weh46ZiOe.rrwKkf8Qm...	0	real
4		monkey 1	Testing@gmail.com	$2a$10$bM/F3STe1jSIRKLHSijLYuHY4oP38f21k...	0	testset
6		monkey 1	teste@gmail.com	$2a$10$o0ibzMwOxFpxpjSoXR/hluiN.UMF453E...	30	|
9		monkey 1	realTest@gmail.com	$2a$10$RWIMB9AMdaFcZXJLt5x2le7wIAowKb...	70	realTest
12		monkey 1	testing123@gmail.c...	$2a$10$Rc8Dn0aXOy2fiu/Cwd310.qtf6UfFJXt9...	0	testing123
13		monkey 1	monkeyFan@gmail....	$2a$10$.1O32esUmwW3IHxyPyNACOhk.C87z...	0	monkeyFan
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 10: The current list of users in Polus' local MySQL server (We can see the account (with user id 13) that was made to get a screenshot of in figure 8).



user_id	completed_question_ids
9	multi-5
10	select-1
10	select-2
10	multi-3
10	multi-4
10	multi-5
10	multi-1
10	multi-2
11	multi-1
11	multi-2
11	select-1
11	multi-3
11	select-2
11	multi-4
11	multi-5
11	multi-6
11	select-3
13	multi-1

Figure 11: Part of the list of completed questions, each entry is linked to a user ID. We can see the example user we made (user id 13) after he answers the first question in lesson one.

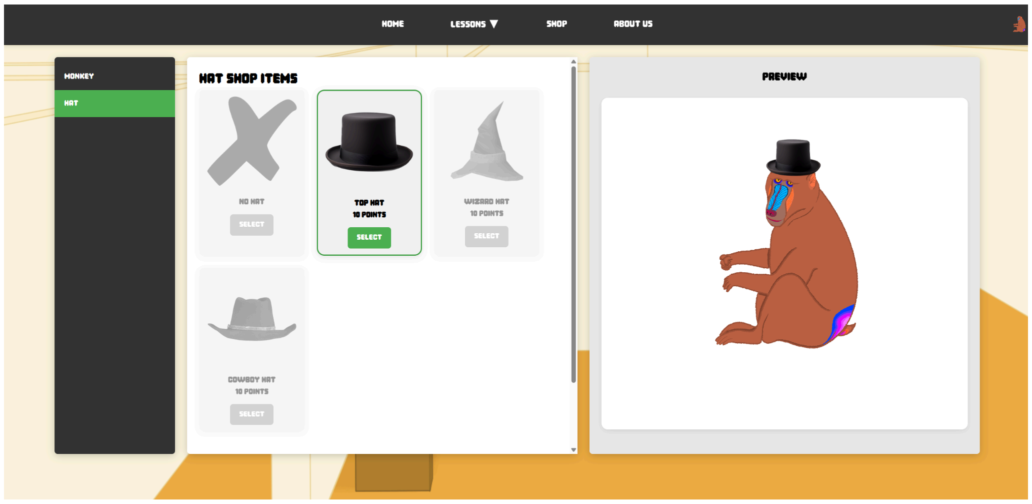
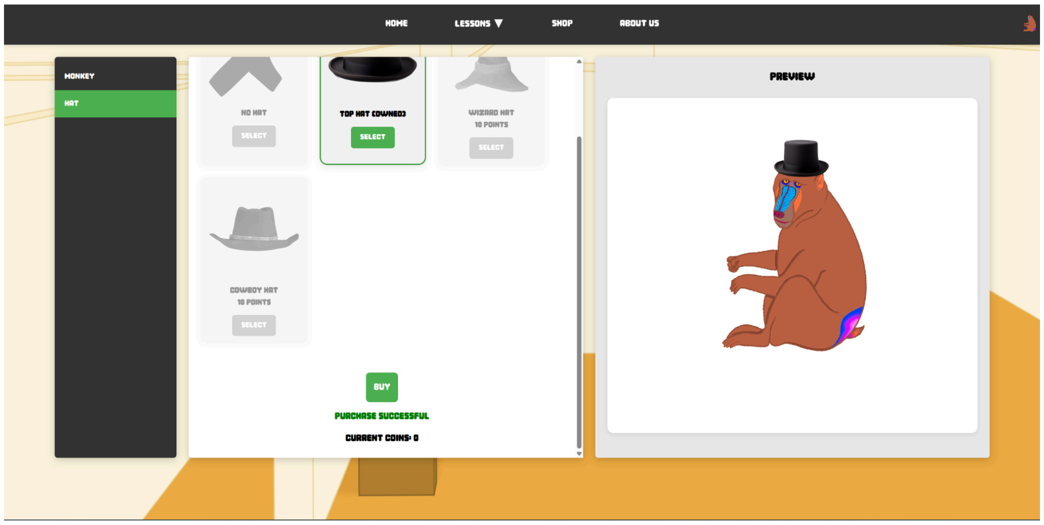


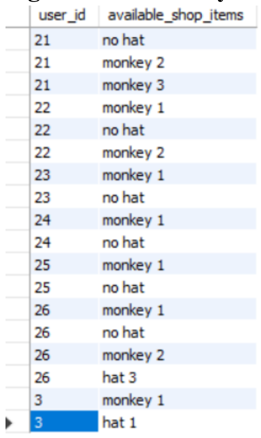
Figure 12: User looking in their shop. They do not have the item unlocked, but can see the current look of their Monkey.



PURCHASE SUCCESSFUL

CURRENT COINS: 0

Figure 13: User buys an item.



user_id	available_shop_items
21	no hat
21	monkey 2
21	monkey 3
22	monkey 1
22	no hat
22	monkey 2
23	monkey 1
23	no hat
24	monkey 1
24	no hat
25	monkey 1
25	no hat
26	monkey 1
26	no hat
26	monkey 2
26	hat 3
3	monkey 1
3	hat 1

Figure 14: Backend reflecting the user (user id 3) buying the item

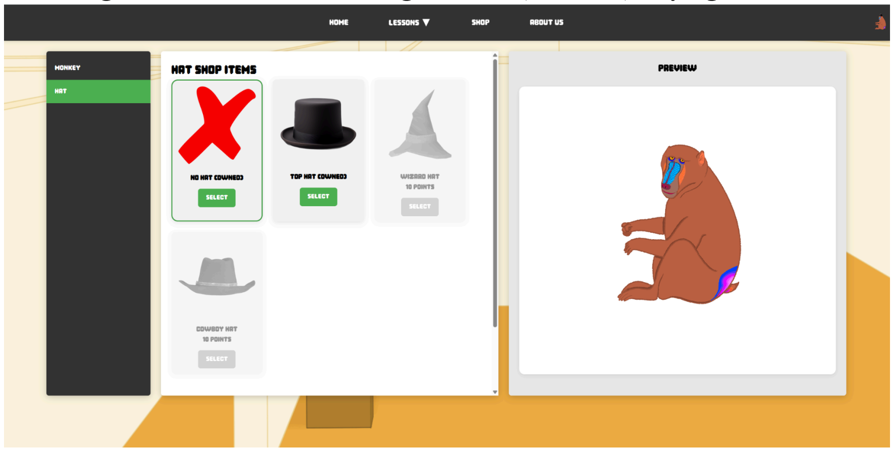
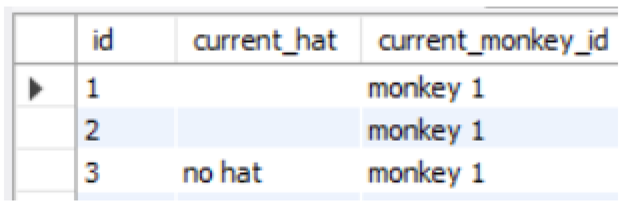


Figure 15: User (user ID 3) switching to having no hat.



id	current_hat	current_monkey_id
1		monkey 1
2		monkey 1
3	no hat	monkey 1

Figure 16: The backend reflects the user's action (User ID 3).