Hello everyone! We are happy to share with you our web application - Friday Night. It is an online game store.

**1.** **Introduction**

**a)** **Problem**

Online games have huge commercial value. Statistics show that there are about 3 billion active game players worldwide. However, there are some pressing issues in this area.

**First**, there are so many online game platforms out there right now. Each has its own unique advantages, deals, and community, which is dizzying.

**Second**, the games on these platforms are usually expensive.

**b) Solution**

Our solution is to build a one-stop online game platform. Friday Night is an online store specialized in selling online games. It has three major features.

**First**, It provides a collection of popular games. So players don’t need to check out different platforms one by one to buy new games.

**Second**, all the products on our website are reasonably priced and affordable for users.

**Third**, it creates a community for players to get the latest news and communicate their gaming ideas.

**2. Architecture and technology stack**

**a) Architecture**

We use model-view-controller architecture pattern. The MVC pattern separates application functionality into three kinds of components.

**Model** contains the core functionality and data.

**View** displays the information to the user.

**Controller** handles the input from user.

**b) Technology stack**

We use MERN stack.

For the **frontend**, we use React, Redux and React Bootstrap. Redux is an open-source JavaScript library for managing and centralizing application state.

For the **backend**, we use MongoDB, node.js, and express.

We also use PayPal **API** to deal with payments.

**3. Data model**

We have four data collections in our project. This graph shows the relationship between them.

The **users** collection is to store users' information.

The **products** collection stores the product information. We also store the reviews here. So the reviews can be easily rendered on the detail page for each product.

In the **orders** collection, an order has reference to the user who placed this order and the product that is contained in this order.

We need to mention this extra **reviews** collection. Because reviews need to be conditionally rendered on the home page. We find it more convenient to have a separate reviews collection so that we can access the data more efficiently.

**4. Demo (Play video)**

Please watch our demo video.

**5. Reflection**

**a) Proud of**

Overall, we are very satisfied with this project.

1. We're proud of our **UI** design, it's dark mode, which is pretty cool.
2. We use React Helmet component to dynamically manage the document’s head section, including setting the title, and meta tags for the document.
3. We strategically use react-bootstrap components to improve page performance. For example, we use **Tab** components on the detail page and profile page to avoid repeated page rendering, thus the page loads faster.
4. We use the state management tool **Redux**, which keeps our code clean and makes the development more efficient.

**b) Challenging**

But using redux results in big trouble when we do the test. Errors happen when we use the regular test methods. Finally, we import redux-mock-store package to solve this problem.

**6. What to do next**

Although this project is completed as a final assignment, we would like to continue to work on it.

First, we want to implement the **administrator** function of this online store. The administrator has the authorization to manage users, products, and orders.

Second, we will optimize the storage of product **images**, because now we just store images in the front end.

Third, we will improve the **search** function. Now it can only search by game name. We will try to implement multi-criteria search and filter.

Last, we also want to implement **pagination** function to improve user experience.

That’s all for our presentation. Thank you!