CSC309 Project Information

Last Updated on Nov 6, 2025

Oct 16, 2025: Project Information Posted

Nov 6: A delay on the release of project grading rubrics

For the term project, you are to develop a web application for the loyal program and connect it to the backend server that you will be building for Assignment 2. The deliverable is a fully-functional website that is shippable, presentable, self-explanatory, smooth, and beautiful. The system will allow users to accumulate and redeem points, while providing role-based access control for cashiers, managers, and superusers. You can find general information about the project and its features from the Assignment 2 handout, and you should continue to refer to it for descriptions of required use cases.

Important Dates

- November 7, 2025: Last day to join a project team by yourself.
- November 8/9: Detailed Rubrics for Project Posted. TAs are assigned to project teams.
- November 24, 2025: Peer Feedback form & First day to start scheduling the grading interviews.
- December 5, 2025: last day to do CSC309 Project Grading Interviews

Setup

1. Log onto MarkUs
2. Go to the **Project repository**. If you are not in a group, please either form your group or join another group by November 7, 2025. Otherwise, we may put you into a random group, or have you do the project by yourself.
3. ONLY ONE PERSON IN THE GROUP SHOULD DO THIS:
    a. Check out the **Project repository** from MarkUs. There should be an empty file and two directories named **frontend** and **backend**. You may add any number of files to both directories, but do not add anything to the root directory.
    b. Copy everything from your **A2** directory into the backend directory in **Project**.
    c. Make sure the backend code is working. Then, commit everything.
4. OTHER GROUP MEMBERS SHOULD DO THIS AFTER STEP 2:
    a. Check out the **Project repository** from MarkUs.

Requirements

You are to implement the frontend of the loyal program using React.js. Your website must be deployed and publicly available. You are not required to consider concurrency or scalability, i.e., it is OK if you continue to use sqlite3 as your backend database, and you may assume only one user will visit your site at any given time.

Documentation

1. **INSTALL**: This document should include a detailed description of any preparation needed for your code to run in a new environment. Include:
    a. Prerequisite packages to install.
    b. How to set up the backend and frontend.
    c. Information on how to deploy your website.

i. The backend server can run on the same (virtual) machine as the web server for the frontend, and is preferred for simplicity.
ii. **Note:** You do not have to share credentials for the hosting space, but you should outline the environment (e.g., a machine running Ubuntu 20.04).

2. **WEBSITE**:
    a. This text file should ONLY contain the **URL** to your publicly hosted website. It should not contain anything else.

Frontend

Place your React project in the frontend directory by running the following command in the frontend directory:
npx create-react-app

You may use React 18 or above, and *you may give this project any name you wish* (please avoid offensive names).

We will not only grade based on functionality, but also on responsiveness, performance, accessibility, etc.

User Interface

Navigation Bar: Ensure different sections of your website are reachable by a navigation bar. This should include a menu of some sort that allows users to view/edit profiles, and log out.
Role-Based Interface: Implement interfaces based on user roles (or rather, the capability of each user)

- **Regular Users**
    - A page that displays the current available points.
    - A page that displays the user's QR code for the purpose of initiating a transaction.
    - A page that allows the user to manually enter a user ID to transfer points
        - QR code scanning capability is NOT required.
    - A page that allows the user to make a point redemption request
    - A page that displays the QR code of an unprocessed redemption request.
    - A page that displays all available promotions.
    - A page that displays all published events.

    - A page that displays a specific event and allows a user to RSVP to an event.
    - A page that displays all past transactions for the current logged in user (with filters, order-by, and pagination).
        - Each transaction card should be displayed "nicely", e.g., instead of relatedId, it should display the utorid of the sender/receiver.
        - Some way to make each transaction type distinct in appearance, e.g., using different colors.
- **Cashiers**
    - A page that allows the cashier to create a transaction.
        - QR code scanning capability is NOT required.
    - A page that allows the cashier to manually enter a transaction ID to process redemption requests.
        - QR code scanning capability is NOT required.
- **Managers**
    - A page that displays all users with filters, order-by, and pagination).
    - A page that allows managers to update users, e.g., make a user verified, promote a user to cashier, etc.
    - A page that displays ALL transactions (with filters, order-by, and pagination).
    - A page that displays a specific transaction, with the option of creating an adjustment transaction for it, and marking it as suspicious.
    - A page that allows managers to create new promotions.
    - A page that displays all promotions (with filters, order-by, and pagination).
    - A page that allows managers to view/edit/delete a specific promotion.
    - A page that allows managers to create new events.
    - A page that displays all events (with filters, order-by, and pagination).
    - A page that allows managers to view/edit/delete a specific event.
    - A page that allows managers to add or remove users from an event.
- **Event Organizer (and all Managers)**
    - A page that displays the events that the user is responsible for.
    - A page that allows the user to view/edit a specific event that he/she is responsible for.
    - A page that allows adding a user to the event that he/she is responsible for.
    - A page that allows awarding points to a single guest, or to all guests who have RSVPed.
- **Superuser**
    - The ability to promote any user to managers or superusers.

Note that the exact number of pages is up to you, as long as the functionality exists. For example, the create and edit event page may look exactly the same, with the exception that the edit event page has input elements pre-filled with existing data.
URL Management: Users should never need to manually change the URL, except for switching roles (e.g., between cashier and manager views). Use React Router to implement page navigation, ensuring the browser's URL changes accordingly. This allows users to navigate with the back/forward button or revisit specific pages via their URLs.

QR Code

You are required to support **displaying** QR code for the following two scenarios:

- A QR code used to identify a user, so that a user can initiate a transfer or purchase transaction.
- A QR code used to identify a redemption request, so that a cashier can process it.

Pagination

Implement pagination for every API that returns a list of objects (e.g., transactions, events, promotions). You can use regular buttons or infinite scrolling to manage large datasets efficiently.

Backend

You are free to make any changes to your backend server to implement the additional features. At bare minimum, you will need to add the cors package to your backend so that your React development server can access the backend.

```
const cors = require('cors');

const app = express();


// Set up cors to allow requests from your React frontend

app.use(cors({

  origin: 'http://localhost:3000',

  methods: ['GET', 'POST', 'PUT', 'DELETE'],

  allowedHeaders: ['Content-Type', 'Authorization'],

  credentials: true
```

}));

To get full marks, your team must implement 1 **meaningful** new functionality. In addition, there will be a small **bonus point** if your team identifies a key feature where your backend could integrate with a third-party service and use their APIs properly.

Pre-populated Database

To help the TA test your website and improve your presentation, you should write a script to seed your database (remember to check-in this script). Include:

- At least 10 users, including at least 1 cashier, 1 manager, and 1 superuser.
- At least 30 sample transactions, with at least 2 of each type of transaction.
- At least 5 events and 5 promotions (enough to trigger pagination).

Also, note the following:

- The data does not have to be original or real, but it should “look real.”
- Ensure your database/media files are not too large to avoid slow load time.
- We will test correct pagination for EVERY page that displays a list of objects, so have enough data to show at least 2 pages, or enough data to trigger infinite scrolling.

Grading

Grading will be done during a grading session, where you will present your project as a team, and the TA will grade your project.

All team members must attend the grading session. Absent members will not receive any marks for the presentation part. We will only accommodate students who have an emergency.

The last day to do the project grading interviews is December 5, 2025.

Individual Contribution

Each team member is expected to contribute fairly to the project. We will assess participation by reviewing commit histories to evaluate individual contributions, and asking detailed technical questions about the code during the interview. While the group

will receive a single mark, individual marks may be adjusted if we find significant disparities in contributions.
We will send out a peer feedback form during the project development.

Preparation

To ensure your work is properly evaluated, practice demonstrating your website before the session. Your interview will conclude exactly at the one-hour mark, and **any features not presented within this time frame will not be graded**.

Rubrics

Rubrics will be posted on November 6, 2025.

Submission

Submit your work to your group's git repository on MarkUs to the **Project** repository. This assignment will be graded manually through an interview session with a TA.

Link to Submission:
https://markus.teach.cs.toronto.edu/markus/courses/91/assignments/944

Academic Integrity

You are permitted to use open-source packages, images and code snippets from the Internet, including AI-generated content (e.g., ChatGPT), as long as you provide proper attribution.

- **Packages**: List any additional packages you use in the INSTALL document and include instructions on how to install and use them.
- **Code Snippets**: If you incorporate external code, cite the source directly as a comment above the relevant code.
- **Assets**: Clearly attribute any external images, fonts, or other assets at the point of import in your source code.

Important: Sharing code with other teams is strictly prohibited. Submitting work that contains identical or highly similar passages of code without proper attribution will be considered an academic offense. Ensure that all external contributions are properly credited to maintaining academic integrity.

If your team uses AI, please indicate how your team used AI, or indicate “Our team did not use any AI for this course project.”
You won’t get a lower mark by using AI, as long as you document the usage properly.

Tips

- While we have not covered React yet and you are working on Assignment 2, it is still good to make use of the time to design the frontend using tools such as Figma.
- When you are assigned to a TA on ~~November 6~~, attend your tutorials, show TAs for your work, your ideas, and get feedback.