Backend

TCX2004

Iteration 1

Project Brief

Design and implement a backend system for a shared expense management platform, modeled after Splitwise. The system should allow users to securely register and authenticate, create and manage expense groups, add and track expenses, record payments, and automatically calculate balances among group members.

Grading (35% of grade)

- **Iteration Grading (21% total):**
  - Iteration 1 is worth **10%** of your final grade.
  - Iteration 2 is worth **10%** of your final grade.
  - Iteration 3 is worth **1%** of your final grade.
  - You will be graded solely on the completion of the required functionality for that iteration.
  - Every team is expected to complete Iteration 1 and 2.
- **Final Submission Grading (14% total):** Your final submission will be graded as follows:
  - **Prototype (7%)** Judged from your code, documentation, and presentation. You must show that **one or two concepts** have been applied from this course for each criterion:
    - Quality of **system architecture and object-oriented modelling**, with clear and appropriate diagrams (e.g., class, sequence, architecture, use case diagrams) that reflect sound design principles (abstraction, cohesion and coupling).
    - Evidence of **testing**, including scripted and/or exploratory testing.
    - Effective use of **version control**, issue tracking, and collaborative development practices, showing clear division of responsibilities and meaningful contributions.
    - **Implementation quality**, including code correctness, reliability, and maintainability, with adherence to coding standards.
    - Identification and mitigation of common **security** risks, with secure coding practices and deployment considerations.
    - Robust **error handling**, ensuring graceful failure and clear feedback for users and developers.
    - **Extensibility** and maintainability, supporting future enhancements and cross-platform deployment.
  - **Usability (3%)** Evaluated based on the number and severity of bugs in the application, as well as the overall user experience, and ease of use.
  - **Documentation (2%)** Clarity, completeness, and succinctness of your `README.md`:
    - Clearly document your application's setup, and deployment steps in your repository.
    - Clearly explain how your implementation meets the Prototype grading criteria.
    - Link to external documentation (e.g., OpenAPI docs) to avoiding duplication.
    - Focus on technical documentation for developers; do not include user-facing guides.
  - **Presentation (2%)** Quality of your 15-minute presentation, upload to Group Project Final Submission on Canvas.
    - Demonstrate your application's key features and workflows.
    - Clearly explain how your implementation meets the Prototype grading criteria.
    - Ensure the presentation is well-structured, concise, and highlights your team's contributions and problem-solving approaches.
- **Contribution:**

- Each team member must contribute meaningfully.
- Peer Evaluations will be conducted.
- Freeriding may result in individual grade adjustments.

Instructions

1. Access and accept the assignment at: https://classroom.github.com/a/_63RRTUw
   - Check your PG number at Canvas Groups; your team number may have changed.
   - **(One person in your team)** When prompted to "Create a new team", use your PG number from Canvas in the format `pg<number>`. For example, for PG 00, use `pg00`.
   - **(Other team members)** Join the team named `pg00`.
   - By accepting the assignment, a repository is created along with a Github team: https://github.com/orgs/tcx2004/teams, which you may use for your project.
2. Your team's VM is at `tcx2004-pg<number>-i.comp.nus.edu.sg`. For example, PG 00 uses `tcx2004-pg00-i.comp.nus.edu.sg`.
   - Default login: `sadm`, password: `ehanadmin`. Change this password immediately after first login.
   - You are **strictly prohibited** from accessing other group's VM. Doing this will constitute cheating and disciplinary actions may be taken.
3. Ensure your application uses FastAPI, HTML/CSS/JS and React Native, any database system, and is fully containerized with Docker on the provided VM. Consult your TA before using other tools/libraries. **Unauthorized tools may result in deductions**.
4. Push all code and documents to your team's GitHub repository.

Submission Procedure

1. Deploy your application to both staging and production environments on your assigned VM.
2. Verify all features by thoroughly testing in both environments to ensure functionality, reliability, and readiness for demonstration.
3. Tag your repository with the `iter1` git tag once your team is ready to submit.
4. Notify your TA to schedule a consultation and prepare a live demo (maximum 15 minutes) showcasing how users complete each user story, emphasizing usability and functionality.
5. Be prepared to discuss your implementation during the demo, as your TA may ask questions.
6. Present your demo over Zoom; the session will be recorded for review.
7. Receive feedback within three days regarding whether your team has met the iteration requirements.

User Stories

Implement the following user stories using only the API (no frontend required), the user must be able to interact with the system using `curl` commands.

- As a new user, I can sign up, log in, and log out using my email so that I can securely manage my account and expenses.
- As a registered user, I can create, view, update, and delete expense groups so that I can organize shared expenses for different contexts.
- As a group admin[1], I can add, update, and remove group members so that I can manage group membership and ensure only relevant people are included.
- As a group member, I can invite new members to join a group and view the current member list so that I can keep group membership accurate and up to date.
- As a group admin, I can view a detailed audit trail of all changes to expenses and balances so that I can ensure transparency and accountability.
- As a group member, I can create, update, and delete expenses (that I have entered), specifying details such as amount, date, payer, and cost-sharing breakdown so that I can accurately record and manage shared costs.
- As a group admin, I can create, update, and delete expenses within the group so that I can manage shared costs and maintain accurate financial records for all members.
- As a group member, I can set up recurring expenses so that I can automatically track regular charges like rent or subscriptions.

[1]Defined as the creator of the expense group.

- As a group member, I can view all expenses in the group so that I can understand shared spending and contributions.
- As a group member, I can rely on automatic balance calculations so that I can see who owes what without manual calculations.