Sprint 1 Retrospective

CS 307 Fall 2022

Team 9 - Purdue Course Finder

Alex Kobus, Alex Plump, Tommy Lane, Peter Zong

What went well

In general,

We successfully initialized and started creating both the frontend and backend of our application. We successfully implemented communication between our frontend and backend, including account authentication, creation, modification, and deletion. We also successfully set up our database that we are using to cache data from the Purdue API as well as store user account information, including security and the hashing of passwords. We successfully retrieved data from the Purdue API and stored it in our database and accessed it from our frontend.

User story 1

As a User, I would like to see information about current Purdue courses.

#	Task Description	Estimated Time	Developer
1	Forward information from Purdue API in backend	4 hours	Tommy
2	Create a prototype UI to display the data for each course	2 hours	Alex K
3	Connect frontend to backend	1 hour	Alex K

Completed:

The backend of our application retrieves course information from the Purdue API and caches it in our database. The frontend displays the data as retrieved from our backend.

User story 2

As a Professor, I would like to see the number of students registered in a section.

#	Task Description	Estimated Time	Developer
1	Create an endpoint for the number of registered students in a section	1 hour	Tommy
2	Create a prototype UI to display the number of students registered in each section of each course	2 hours	Alex K
3	Connect frontend to backend	1 hour	Alex K

Completed:

The backend of our application retrieves section registration information from the Purdue API given a course and caches the data in our database. The frontend displays the data as retrieved from our backend.

User story 3

As a User, I would like a dedicated page for signing up.

#	Task Description	Estimated Time	Developer
1	Initialize React Project & setup packages	2 hours	Alex K
2	React Tutorials and setup local React environments	3 hours (each)	Peter, Tommy, Alex P
3	Lay out and finalize page design	1 hour	Alex K
4	Create signup page UI	3 hours	Alex K
5	Connect input fields and buttons to send credentials to server	3 hours	Alex K
6	Implement email/password format checks	2 hours	Alex K
7	Setup placeholder actions depending on server login response	2 hours	Alex K
8	After user story 5 is done, connect server to database (user credentials)	2 hours	Tommy
9	Create unit tests and perform manual tests to ensure correct page functionality	3 hours	Alex P

Completed:

To create an account, users can go to the completed signup page. The necessary format checks, including blank fields, invalid email syntax, and unmet password complexity requirements have been implemented. Other error checks, such as if an email is already in use or if the password and confirm password fields don’t match are also implemented. Upon successful account creation, a success popup is shown.

User story 4

As a User, I would like a dedicated page for logging in.

#	Task Description	Estimated Time	Developer
1	Lay out and finalize page design	1 hour	Peter
2	Create login page UI	3 hours	Peter
3	Setup placeholder actions depending on server login response	2 hours	Peter
4	After user story 5 is done, connect server to database to check user credentials	2 hours	Peter
5	Create unit tests and perform manual tests to ensure correct page functionality	3 hours	Alex K

Completed:

After creating an account, users can go to the completed login page to log in to their account. Upon successful login, users are redirected back to the home page and their login session is stored in an authentication token to keep them logged in until they exit the site. Upon unsuccessful login, an error message is displayed.

User story 5

As a Developer, I would like to store user account information in a SQL database.

#	Task Description	Estimated Time	Developer
1	Setup initial Spring Boot backend configuration	2 hours	Tommy
2	Spring Boot tutorials and setup local Spring Boot environments	3 hours (each)	Alex K, Alex P, Peter
3	Docker tutorial, testing, prototyping	4 hours (each)	Alex P, Alex K, Peter
4	Sample various databases within Docker Containers	3 Hours	Alex P
5	Setup database as Docker container	3 hours (each)	Tommy, Alex P
6	Integrate Spring Boot back-end with the Database	4 hours	Tommy
7	Define database schema in backend	4 hours	Tommy
8	Learning SQL Format, Commands: Selecting, Adding, Modifying, and Removing entries.	2 hours (each)	Alex P, Alex K, Peter
9	Adding and removing items from Database using Backend	3 hours (each)	Tommy, Alex P
10	Containerize the backend using Docker	3 hours	Tommy
11	Create docker compose file	2 hours	Tommy

Completed:

The backend and database were successfully configured. Things were done to ensure the uniformity of the application across our local environments as well as in preparation for production. The database we went with was PostgreSQL which we depended on for many other tasks in this sprint. The database is run from a Docker container to ensure it operates the same on each of our local environments. A docker-compose.yml file was also created so that the entire application could be run in a production environment as a test.

User story 6

As a User, I would like to see a tutorial that explains how to use the software.

#	Task Description	Estimated Time	Developer
1	Lay out and finalize page design	1 hour	Peter
2	Create page UI	2 hours	Peter
3	Add UI Mockups from the Design Document and describe how to use them	1 hour	Peter
4	Add images of the completed pages for logging in and creating accounts and describe how to use them	1 hour	Peter

Completed:

The completed tutorial page details, with images, instructions on how to use our website. The table of contents at the top help users quickly find the information they want, smooth scrolling down the page. For pages and features not currently implemented, some UI mockups are set in place with basic instructions with more details to be added once implemented.

User story 7

As a User, I would like to delete an account.

#	Task Description	Estimated Time	Developer
1	Create a page for account deletion	1 hour	Alex P
2	Add buttons and pop-up when pressed	2.5 hours	Alex P
2	Search and delete account credentials from database	1.5 hours	Alex P
3	Unit tests	1 hour	Alex P

Completed:

The account deletion page has its own UI that displays buttons to return to the menu page. It also displays the current user's account data on the screen to them. When the user wants to delete their account, they press the delete account button and confirm on the popup. The client then sends the server a request to delete the account associated with the logged in email, and the credentials are cleared from the database. Additionally, we implemented unit tests for this page which confirm correct loading and configuration upon rendering.

User story 8

As a User, I would like to change my account email and password.

#	Task Description	Estimated Time	Developer
1	Create a page for account modification	2 hours	Peter
2	Find and change database email/password entry	2 hours	Peter
4	Unit tests	1 hour	Peter

Completed:

The completed account modification page allows a logged in user to change their email and/or password. The new email/password must meet the same syntax and complexity requirements as when signing up for an account, and changing emails to one that already exists is not allowed. After changing the information, the next time the user logs in, they need to use the new login info. Additionally, we implemented unit tests for this page which confirm correct loading and configuration upon rendering.

User story 9

As a Developer, I would like to save encrypted account login information in a database.

#	Task Description	Estimated Time	Developer
1	Learn and employ encryption to client-side message passing	4 hours	Tommy
2	Test to ensure data is properly encrypted and saved	1 hour	Tommy

Completed:

The backend uses TLS to secure interactions from the frontend. This was implemented locally using certificates created by mkcert. The user account system also stores sensitive credentials (the passwords) as hashes to prevent the passwords from being directly leaked if the database contents were to be lost.

User story 10

As a Developer, I would like to save backend logs of what pages are accessed most frequently.

#	Task Description	Estimated Time	Developer
1	Log each page visited by users to the server	2 hours	Tommy
2	Save logs to a file	2 hours	Tommy
3	Test that logs are being generated and saved correctly	1 hour	Tommy

Completed:

We implemented Flogger/SLF4J into the backend to handle logging. Specifically, a filter was introduced to log each request received from a client. These logs were configured to save to a file locally as well which updates as the backend runs.

What did not go well

"Given the account information is properly encrypted, when viewing the emails stored in the database, then we should not be able to tell what the original emails are."

The Acceptance Criteria above was a mistake on our part. We in fact did not want to encrypt emails. This was so that each user's email could be displayed on both the modify account page and the delete account page. We purposely left this out of our design because it would cause issues for areas other than logging in, which is what we only anticipated emails being used for.

Testing

In the first sprint, frontend testing setup was much more difficult and time consuming than anticipated. As a result, most of our tests were manual tests rather than coded unit tests, possibly missing some edge cases. We hope to improve both testing types in future sprints.

How should we improve?

- We will take more care to ensure that all our User Stories, Tasks, and Acceptance Criteria in future sprints are accurate to what we plan to do with our project, and update them during the sprint if necessary.
- Now that we have the packages installed to do automated testing of React components and JavaScript code, it will be easier for us to create unit tests for our frontend to ensure that it is working properly.
- Now that we all have the environments setup and some familiarity with the frameworks that we are using for this project, we should be better at making good time estimates in our sprint planning documents and hopefully have less of a need to modify them during the sprint.
- We did not start thinking about Sprint 2 during our first sprint. In future sprints, we will employ flexibility in our code so that building off what we have already developed will be less time-consuming.