**Overview & Timeline**

Learning new terminologies is a big part for both students and researchers when they study a certain field. However, it usually takes a relatively long time for people to search terminologies, choosing the appropriate definition, and making flashcards for the terms they want to memorize. Storch is an effective study software that can automatically analyze uploaded study materials and generate flashcards for the terminologies that appear in certain study materials. After users upload their study materials in pdf, video, images and so on, Storch is able to analyze the study material **utilizing some pdf parsing packages and some text recongnizer APIs**, search the terminologies that appear in that material **using a natural language processing package for javascript and some “algorithms”**, and ask the user if they want to generate the flash card. The flashcards contain a well-defined definition of the specific terminology based on the subject from Wikipedia and other reliable online dictionaries. Users are able to save the flashcards in history and review them later. Besides, users can customize the way Storch generates the flashcards by setting the level of difficulty and the language they prefer. If the user wants to know more about the subject, they can search the flashcards with specific tags and view the **flashcards created or generated from other user’s uploads**. In addition, **there is one specific type of document we want to treat with special care: the syllabus.** Storch will add important events and deadlines shown on the syllabus and users can export to Google calendars.

Specs:

* React.js in javascript
* Node.js in javascript
* MySQL using SQL
* A lot of help from third-party APIs, including Microsoft Azure, like text recognizations and NLP and vocabulary lookup
* Deployment on Digital Oceans or other cloud platforms

Why good:

* Open source
* Students were driven, not for profit
* Easy to use due to the automated flashcard generation
* Featured on sharing and learning, promoting a healthy community

Timeline:

|  |  |
| --- | --- |
| Date / Month | Activity |
| 5/20 - 6/20 | * Learn React.js, Node.js, MySQL, find suitable APIs * Make prototypes either using figma or Adobe XD * Check if the packages listed match the actual needs of our app * If the APIs do not match our needs, either find alternatives or redesign the app * Finalize the first draft of a database schema * Come up with drafts of user case tests * Reiterate and finalize mockups |
| 6/20 - 7/10 | * Complete coding backend database (basic CRUD without interacting with third-party APIs) * Complete and test home page * Complete and test login page * Setup and test the basics, including domains, SSL, auths, API keys for the APIs we are using and so on |
| 7/10 - 8/10 | * Complete and test backend interactions with third party APIs, such as the calls to look up terms and fetch content and parse the results * Check if the existing database schema still works, fix if not * Complete and test the rest of the front-end with mocking data |
| 8/10 - 8/15 | * Recheck if our backend API matches the requirements of our front-end * Reiterate and redesign if needed |
| 8/15 - 9/15 | * Complete and test the communications between the front-end and back-end * Write UI-tests that tests complete user-interaction scenarios |
| 9/15 - 10/1 | * Release to a small group of users and receive feedback * Change and improve based on the feedback |
| 10/1 - | * Deploy to production-level environments * Publish the complete app * Monitor the third party API usages and upgrade the plans if needed * Calculate the costs and adjust (switch to cheaper APIs) is needed |

Next Steps:

* Improve the community feature
  + Users will be interacting with each other more
  + Like customizable profile pages and leader board system
  + Professors and other teaching team members will be able to post their customized contents
* Improve the note taking feature
  + Will support a built-in editor where user can take notes directly
  + And will attempt to generate the flashcards on the fly when the user is typing
* Add recommender system
  + Will recommend the documents or flashcards that users are possibly interested in based on their historical data
* Reduce the costs or find a way to balance the cost
  + If we get too many users, the free plans of the third party APIs will not be available anymore
  + Find a way to balance the costs to maintain the project