MAD-II Project : Final Report

Date: 28 March 2022

Author

Tushar Shrikrishna Supe
21F1003637
21f1003637@student.onlinedegree.iitm.ac.in

I hold a BTech degree in Mechanical Engineering from VNIT Nagpur. I have worked in RIL for a year but as I am interested in AI & ML, I joined this BSc Degree in Programming and Data Science offered by IIT Madras.

Description

In this **Flashcard Web App** project we need to create a web based app using Flask, Python, and Vue based UI which follows basic MVP design and Client–server model. Client can create deck and cards and review deck. The app also has advance features like import/export deck, generate report. All page rendering and UI functions are handled by Vue (UI).

Technologies used

Technology	**Purpose**
Python	Core programming language of this app
HTML	Creating web pages
CSS	Styling web pages
JavaScript (Vue)	For basic functionalities like loading templates, functions, routing to pages, fetch requests.
flask	Web framework library
flask-sqlalchemy	Library to create SQLite database, to connect with database
Flask security	Library for JWT token based authentication
flask-restful	Library for implementing CRUD API for deck management
Jinja2	Library for rendering main html template at the start of the app
requests	Library to send get, post, put, delete requests to API
pandas	Library to create/read/write CSV file
weasyprint	To convert jinja html template into pdf format
Smtp	To sent emails via python
celery	To create and manage async jobs and scheduled jobs
redis	Redis database for caching and backend async jobs
mailhog	To create dummy mail server for testing purposes
*in-built: os, time, datetime, json, random, pyjwt	
*basic: device with web browser and internet connection	

DB Schema Design

Table name: users			
Column	**Datatype**	**Constraint**	**Description**
userid	Integer	Primary key, Autoincrement	Unique user id for every user
username	String	Unique, Not null	Unique user name (can't be edited once created)
email	String	Unique, Not null	Unique user email (for login purpose)
password	String	Not null	Password specified by user

Table name: deck			
Column	**Datatype**	**Constraint**	**Description**
deckid	Integer	Primary key, Autoincrement	Unique deck id for every deck
deckname	String	Not null	Unique deck name for same user id
deckdesc	String	Not null	Description of the deck
ltime	Integer		Last reviewed time of deck (time as timestamp)
lscore	Float		Last reviewed score of the deck
oscore	Float		Overall score of the deck
userid	Integer	ForeignKey("users.userid"), Not null	To link user to deck

Table name: cards			
Column	**Datatype**	**Constraint**	**Description**
cardid	Integer	Primary key, Autoincrement	Unique card id for every card
question	String	Not null	Front side of card, must be unique per deck
answer	String	Not null	Back side of the card
ltime	Integer		Last reviewed time of card (time as timestamp)
lscore	Float		Last reviewed score of a card
deckid	Integer	ForeignKey("deck.deckid"), Not null	To link card to deck

Single database is used to store user details, decks and cards to reduce complexity while managing decks and cards. Userid from user table links to deck table and deckid links to cards table so for logged-in user other decks & cards are not accessible which provides data security. Both User-Deck and Deck-Cards follows one-many relationship (for same user two deck can have similar cards but as per this db schema two cards will have different cardid). Also one-many relationship for deck-card is defined in ‘models.py’ so given a deckid all the cards in that deck can be retrieved. Along with this, ‘on delete cascade’ constraint set on deckid so that deleting a deck will delete all its card records from cards table also.

API Design

The CRUD API is implemented on both Deck and Cards table. Once main html is rendered all the frontend is handled by Vue (SPA). Vue sends fetch requests to API and python manages the database using sqlalchemy. Vue displays response messages to users. All requests are sent with JWT token in the header.

Resource	Request	Endpoint	Request body	Purpose (for current user)
Deck	GET	/api/deck	-	To get details of all decks
Deck	POST	/api/deck	deckname, deckdesc	To create a new deck
Deck	GET	/api/deck/<int:deckid>	-	To get deck details
Deck	PUT	/api/deck/<int:deckid>	deckdesc	To update deck description
Deck	PUT	/api/deck/<int:deckid>	ltime, lscore	To update last reviewed time, score of deck
Deck	DELETE	/api/deck/<int:deckid>	-	To delete a deck
Deck	GET	/api/deck/<string:deckname>	-	To get deckid from deckname
Cards	GET	/api/deck/<int:deckid>/card/<int:cardid>	-	To get single card details
Cards	PUT	/api/deck/<int:deckid>/card/<int:cardid>	question, answer	To update front and back details of a card
Cards	PUT	/api/deck/<int:deckid>/card/<int:cardid>	ltime, lscore	To update last reviewed time and score of a card
Cards	DELETE	/api/deck/<int:deckid>/card/<int:cardid>	-	To delete a card
Cards	POST	/api/deck/<int:deckid>/card	question, answer	To create a new card
Signup	POST	/signup	User details	To validate data and create user account
Login	POST	/login	User details	To log in user and provide a JWT

Architecture and Features

This project implements MVP architecture. Model is python-flask with sqlite database. View is what user see and presenter is frontend Vue which interacts with user as well as sends fetch requests to API for CRUD. Vue acts as a “middle man”.

Features:

- ✓ Basic UI with Vue (it’s a SPA)
- ✓ Deck and card management (CRUD) with RESTful api
- ✓ User login and all api request with JWT token authentication
- ✓ Daily reminder with Google Chat webhook and email alerts
- ✓ Monthly report generation and emailing
- ✓ Import and export decks (.csv format)
- ✓ Caching basic data like user details, tokens

Video

Link: https://drive.google.com/file/d/1XNXoBwlxghFL_-2TYnJyYfepcrmJF1mf/view?usp=sharing