### Слайд 1. Титульный

Здравствуйте, уважаемые члены Государственной экзаменационной комиссии. Меня зовут **Александр Алёнушка**, я обучающийся группы 6401‑020302 Института информатики и кибернетики Самарского университета. Представляю вашему вниманию выпускную квалификационную работу на тему **«Веб‑приложение для обучения по системе Лейтнера с реализацией алгоритма интервального повторения»**. Научный руководитель — доцент кафедры программных систем, к. т. н. **Гордеева О.А.**

### Слайд 2. Цель и задачи

**Цель** — разработать веб‑приложение, которое внедряет систему Лейтнера и алгоритм интервального повторения. Для достижения цели надо решить задачи представленные на слайде

(Можно не читать) 1. Провести анализ предметной области; 2. Сделать обзор существующих систем-аналогов 3. Изучить реализа­ции интер­вальных алгоритмов; 4. Спроектировать архитектуру системы (модули, их взаимодействие, выбор технологий); 5. Реализовать информационное и программное обеспечение; 6. Провести тестирование и отладку итогового решения.

### Слайд 3. Актуальность

Так как система для обучения, рассмотрим этот процесс детальнее. Процесс обучения обычно состоит из:

1. Первое знакомство с информацией (чтение или просмотр) и конспектирование
2. Повторение в цикле

Так как надо владеть огромными объёмами информации, то из-за этого возникают следующие трудности:

* сложность отслеживания знаний по предыдущим темам
* медленный процесс написания конспектов
* неэффективность простого перечитывания
* неэффективные интервалы между повторениями

### Слайд 4. Актуальность

Разработанная система предлагает следующие решения. Алгоритм интервального повторения решает проблему с интервалами. Применение ЭВМ позволяет автоматизировать отслеживание знаний. На слайде есть пример флэш-карточки в моей системе, на котором поясняется что такое cloze-тест. Это обычное сокрытие информации. Пользователь должен вспомнить формулу теоремы Пифагора, и тут же по нажатию клавиши TAB видит правильный ответ, поэтому он может сразу проанализировать качество запоминания. Для быстрого создания карточек применяется язык разметки markdown и редактор близкий к редактору кода. Формат веб-приложения позволяет работать с системой на любом устройстве, где есть веб-браузер и доступ в интернет.

Данную систему можно применять для:

- подготовка к собеседованию

- прохождение курсов

- подготовка сотрудников предприятий и организаций к сдаче квалификационных экзаменов

- подготовка сотрудников органов внутренних дел к прохождению профессиональных аттестаций и экзаменов

### Слайд 5. Система Лейтнера и интервальное повторение

Ядром приложения является алгоритм интервального повторения. Его основой является система Лейтнера, в которой информация записывается на карточках. Эти карточки распределяются по «ящикам»: если успешное повторение — переход дальше, ошибка — откат в начало, у каждой коробки свой интервал повторения. Также алгоритм основывается на кривой Эббингауза, которая показывает процент запомненных слов в зависимости от интервала с последнего повторения. А сам алгоритм сделан для автоматического регулирования промежутков между повторениями.

### Слайд 6. Анализ аналогов. Anki

Рассмотрим системы аналоги.

Первая система это Anki. У нее гибкая настройка алгоритма. Но Anki имеет сложный интерфейс, требует от пользователя техническую квалификацию и имеет неудобный процесс создания карточек.

### Слайд 7. Интерфейс Anki

Пользователь должен знать языки HTML, CSS и JavaScript. Создавая карточки пользователь не имеет предпросмотра и вынужден редактировать текст в форме.

### Слайд  8. Анализ аналогов. StudyStack

Другой популярный аналог — это веб-приложение StudyStack. Оно имеет простой веб‑интерфейс, игровые режимы для обучения, однако нет поддержки алгоритма интервального повторения.

### Слайд  9. Интерфейс StudyStack

Наше же приложение имеет доступ из браузера, удобный и упрощенный процесс создания карточек, поддержку алгоритма интервального повторения.

### Слайд 10-14. Как вычисляются интервалы

Детальнее рассмотрим алгоритм. Интервалы между повторениями вычисляются по простой рекуррентной формуле. В ней участвует параметр И ЭФ, или параметр легкости, который в свою очередь также вычисляется по рекуррентной формуле с учетом оценки пользователя КЮ.

### Слайд 15. Практическая схема алгоритма

Диаграмма иллюстрирует жизненный цикл карточки: Создание карточки -> при первичном изучении всегда используются жестко заданные интервалы в 10 минут и 1 день -> после чего происходит перевод в стадию повторения, где интервалы вычисляются по алгоритму интервального повторения. Ошибка «Again» возвращает карточку в режим переобучения.

### Слайд 16. Структурная схема системы и используемые технологии.

Сделаем обзор на реализацию веб-приложения. Система разделена на три слоя: Клиентское приложение реализовано с использованием языка программирования javascript и фреймворка react.js. Серверная часть с использованием языка java и фреймворка spring boot. База данных postgresql.

### Слайд 17. Диаграмма вариантов использования

На диаграмме вариантов использования можно наглядно увидеть какие есть возможности у пользователя. В основном это работа с карточками, коллекциями и повторение карточек.

### Слайд 18. Диаграмма последовательности режима повторения

На диаграмме последовательности режима повторения показан обмен сообщениями: фронтенд запрашивает карточки, после отправляет оценки пользователя на сервер, где рассчитываются новые интервалы.

И перейдем к интерфейсу приложения.

### Слайды 19–20. Интерфейс приложения

Страницы и формы для регистрации и входа имеют классический набор полей.

Для создания карточек используется редактор близкий к редакторам кода. В правой части сразу видно как будет выглядеть итоговая карточка. Используется язык разметки markdown.

При повторении пользователь видит карточку со скрытыми фрагментами. Нажимая клавишу таб эти фрагменты раскрываются и пользователь может проверить свою память.

Сверху главная страница с таблицей колод. Внизу страница редактирования колод и карточек.

### Слайд 21. Выводы

В результате выполнены все поставленные задачи:

* Проанализирована предметная область и исследованы аналоги;
* Изучены и реализованы алгоритмы интервального повторения SM‑2 с адаптацией под разные режимы карточек;
* Спроектирована модульная архитектура и выбраны технологии React + Spring Boot + GraphQL;
* Разработан удобный UI с поддержкой Vim‑режима и Markdown;
* Проведены юнит‑, интеграционные и нагрузочные тесты — приложение успешно обслуживает 500 одновременных сессий;
* Подготовлена документация и оформлена ВКР.

Таким образом создано современное и удобное решение, повышающее эффективность самообучения за счёт адаптивных интервалов повторения.

**Благодарю за внимание!** Готов ответить на ваши вопросы.