

**Тема: Уеб приложение за водене на  
дневник с възможност за записване и  
управление на бележки, реализирано с  
използване на облачните услуги AWS S3,  
Amazon Cognito, AWS API Gateway, AWS  
Lambda и Amazon DynamoDB**

Предмет: Приложно-програмни интерфейси за работа с  
облачни архитектури с Амазон Уеб Услуги (AWS)

Изготвил: Йоанна Миленова Ненкова, фн: 7MI0800263,  
имейл: anni.nenkova@gmail.com

Лектор: проф. д-р Милен Петров, година: 2025

# Съдържание

<b>1</b>	<b>Условие</b>	<b>3</b>
<b>2</b>	<b>Въведение</b>	<b>3</b>
<b>3</b>	<b>Теория</b>	<b>4</b>
3.1	Amazon S3 (Simple Storage Service) . . . . .	4
3.2	Amazon Cognito . . . . .	5
3.3	API Gateway . . . . .	5
3.4	AWS Lambda . . . . .	5
3.5	Amazon DynamoDB . . . . .	5
<b>4</b>	<b>Използвани технологии</b>	<b>5</b>
<b>5</b>	<b>Инсталация и настройки</b>	<b>6</b>
<b>6</b>	<b>Кратко ръководство за потребителя</b>	<b>12</b>
6.1	Регистрация и потвърждение на акаунт . . . . .	12
6.2	Вход в системата . . . . .	13
6.3	Работа със записи в дневника . . . . .	13
6.4	Изход от системата . . . . .	15
<b>7</b>	<b>Примерни данни</b>	<b>15</b>
<b>8</b>	<b>Описание на програмния код</b>	<b>15</b>
8.1	Модул за автентикация (Cognito User Pool) . . . . .	16
8.2	Модул за управление на записи (AWS Lambda функции) . . . . .	16
8.3	Модул на клиентската част (React приложение) . . . . .	17
<b>9</b>	<b>Приноси на студента, ограничения и възможности за бъдещо развитие</b>	<b>17</b>
<b>10</b>	<b>Какво научих</b>	<b>18</b>
<b>11</b>	<b>Списък с фигури и таблици</b>	<b>19</b>
<b>12</b>	<b>Използвани източници</b>	<b>19</b>

# 1 Условие

Разработка на уеб приложение, базирано на облачната платформа Amazon Web Services (AWS), което представлява дигитален дневник. Системата трябва да позволява на потребителя да се регистрира, да създава, редактира и трие дневникови записи, както и да преглежда стари такива.

Потребителят преминава през следните основни стъпки:

1. Регистрация и потвърждение на акаунт чрез имейл.
2. Вход в системата с валидни идентификационни данни.
3. Създаване на нов запис в дневника със заглавие и съдържание.
4. Преглед на списък с предишни записи.
5. Редактиране или изтриване на съществуващи записи.

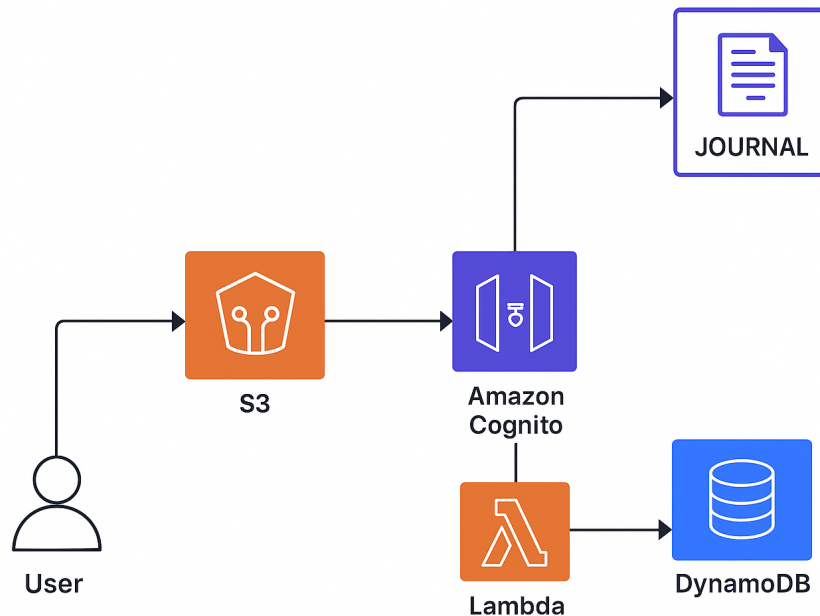
# 2 Въведение

В ерата на облачните технологии и сървърлес архитектурите, изграждането на уеб приложения става все по-гъвкаво, бързо и икономически изгодно. Настоящият проект представлява уеб базиран дневник, изграден върху AWS, който позволява на потребителите да записват важни събития и мисли, достъпни от всяко устройство с интернет връзка.

Основните функционалности включват:

- Сигурна регистрация и автентикация на потребители чрез Amazon Cognito.
- Създаване, редактиране и изтриване на дневникови записи.
- Съхранение на данните в Amazon DynamoDB.
- API слой за управление на данните, реализиран чрез AWS API Gateway и AWS Lambda.
- Хостване на клиентската част в Amazon S3.
- Защита на API методите с потребителска автентикация и контрол на достъпа.

Проектът е изграден изцяло върху сървърлес (serverless) технологии на AWS, което гарантира висока мащабируемост, надеждност и оптимизация на разходите, тъй като се плаща само за реално използваното изчислително време, без да се налага поддръжка на сървъри.



Фигура 1: Архитектура на приложението в AWS

## 3 Теория

Архитектурата на проекта се базира на ключови сървърлес услуги от AWS, които позволяват изграждането на модерни, скалируеми и ефективни приложения без необходимост от управление на сървърна инфраструктура.

### 3.1 Amazon S3 (Simple Storage Service)

**Amazon S3** е услуга за обектно съхранение, която се използва за хостинг на статичните файлове на нашето React уеб приложение (HTML, CSS, JavaScript, изображения). Конфигуриран като уебсайт, S3 bucket служи като начална точка за потребителите, доставяйки бързо и надеждно фронтенд частта на приложението.

## 3.2 Amazon Cognito

**Amazon Cognito** предоставя цялостно решение за управление на потребителски идентичности. Чрез него се реализира сигурната регистрация, вход и управление на потребителски сесии. Cognito се интегрира директно с API Gateway, за да защитава достъпа до бекенд ресурсите, като гарантира, че само автентифицирани потребители могат да извършват определени операции.

## 3.3 API Gateway

**API Gateway** действа като "входна врата" за всички заявки от клиентското приложение към бекенда. Услугата управлява създаването, публикуването и защитата на RESTful API. Тя приема HTTP заявките, валидира ги (често в комбинация с Cognito), и ги пренасочва (routing) към съответната AWS Lambda функция за обработка.

## 3.4 AWS Lambda

**AWS Lambda** е сърцето на сървърлес архитектурата. Това е изчислителна услуга, която изпълнява код (в нашия случай Node.js) в отговор на събития, каквито са API заявките от API Gateway. Вместо постоянно работещи сървъри, Lambda функциите се стартират само при нужда, обработват заявката и спират, което води до изключителна ефективност на разходите.

## 3.5 Amazon DynamoDB

**Amazon DynamoDB** е бърза и гъвкава NoSQL база данни, идеална за сървърлес приложения. Нейната схема "on-read" позволява съхранението на разнородни данни без фиксирана структура. В проекта се използва за съхранение на информация за записи в дневник. Гарантира ниска латентност и автоматично мащабиране.

# 4 Използвани технологии

- **Amazon S3 (Simple Storage Service)** - Обектно хранилище, използвано за хостинг на статичните файлове на React приложението (CSS, JavaScript).
- **Amazon Cognito** - Услуга за управление на потребителски идентичности и автентикация. Осигурява сигурна регистрация и вход в системата.

- **Amazon API Gateway** - Управлявана услуга за създаване, публикуване и защита на RESTful APIs. Служи като входна точка за всички заявки от фронтенда към бекенда.
- **AWS Lambda** - Serverless compute услуга, която изпълнява бекенд логиката (Node.js 18.x), без да е необходимо да се управляват сървъри.
- **Amazon DynamoDB** - Бърза и гъвкава NoSQL база данни за съхранение на информация за потребители и техните записи в дневник.

## 5 Инсталация и настройки

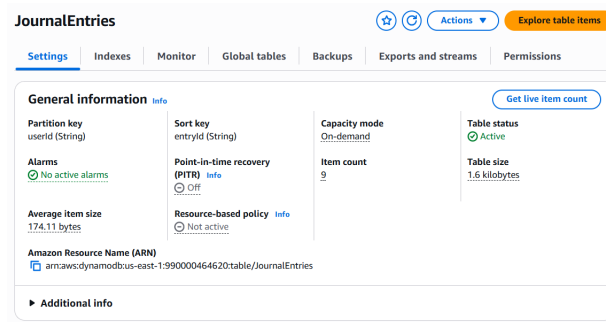
**Стъпка 1.** Конфигуриране на DynamoDB таблица за съхранение на данните. Използваме DynamoDB, защото предлага автоматично мащабиране, ниска латентност и лесна интеграция със сървърлес приложения. За разлика от релационни бази като RDS, които изискват управление на сървъри и фиксирани схеми, или документоориентирани бази като MongoDB, които изискват сами да настроите кластер и резервни копия, DynamoDB е напълно управлявана и гарантира надеждност и висока производителност дори при голямо натоварване.

### 1.1. Създаване на таблицата.

Table name: JournalEntries,

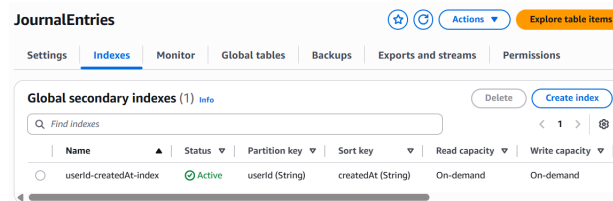
Partition key: userId (Тип: String) - тук ще пазим уникалния идентификатор на потребителя от Cognito,

Sort key: entryId (Тип: String) - това ще бъде уникален ID за всеки запис в дневника.



Фигура 2: JournalEntries Table в AWS Management Console

**1.2.** Създаване на Global Secondary Index за извличане на записи по дата.  
 GSI name: userId-createdAt-index,  
 Partition key: userId (Тип: String),  
 Sort key: createdAt (Тип: String) - тук ще пазим дата и час на създаване на записа.



Фигура 3: JournalEntries Indexes в AWS Management Console

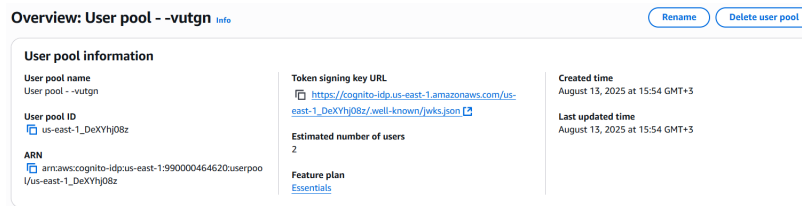
**Стъпка 2.** Конфигуриране на Cognito User pool и App client за автентикация на потребителите.

**2.1.** Създаване на Cognito User pool-a. Изисква имейл за регистрация.

User pool name: User pool - -vutgn,

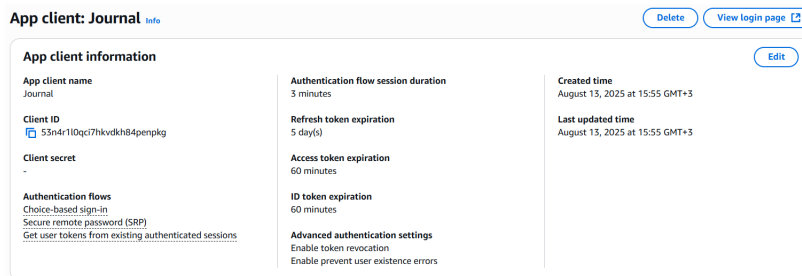
Region: us-east-1,

User pool ID: us-east-1DeXYhj08z.



Фигура 4: User pool в AWS Management Console

**2.2.** Създаване на App client. Представява Single Page Application.  
Client name: Journal,  
Client ID: 53n4r1l0qci7hkvdkh84penpkg.



Фигура 5: App client в AWS Management Console

**Стъпка 3.** Създаване на Lambda функции за управление на записите. Всяка Lambda функция е микросервиз, който отговаря за конкретна операция с данните в DynamoDB. Функциите обработват HTTP заявки от API Gateway, взаимодействат с базата данни и връщат резултат. Те включват логика за валидация на входящите данни, обработка на грешки (например, записът не е намерен или потребителят няма права за достъп) и форматиране на отговора, за да бъдат коректно обработени от фронтенда.

**3.1.** Функция `getEntries` за извличане на записи по дата на създаване. Тя прави заявка към DynamoDB, използвайки Global Secondary Index (`userId-createdAt-index`), за да върне всички записи на конкретен потребител, сортирани по дата на създаване.

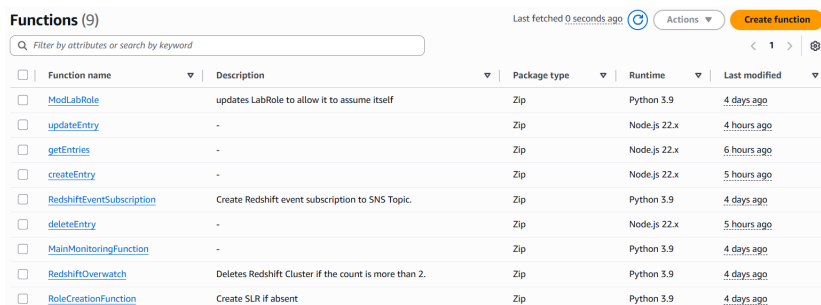
**3.2.** Функция `createEntry` за създаване на запис. Тя получава данни (заглавие, съдържание) от заявката, генерира уникален ID за записа и го съхранява в DynamoDB.

**3.3.** Функция `editEntry` за редактиране на запис. Тя приема ID на запис и нови



данни, след което обновява съществуващия запис в DynamoDB, като проверява дали потребителят е собственик на записа.

**3.4. Функция `deleteEntry` за изтриване на запис.** Тя премахва запис от DynamoDB въз основа на подаден ID, отново след проверка за правата на достъп на потребителя.



The screenshot shows the AWS Lambda console with a list of 9 functions. The functions are listed in a table with columns for Function name, Description, Package type, Runtime, and Last modified. The functions include `ModLabRole`, `updateEntry`, `getEntries`, `createEntry`, `RedshiftEventSubscription`, `deleteEntry`, `MainMonitoringFunction`, `RedshiftOverwatch`, and `RoleCreationFunction`.

Function name	Description	Package type	Runtime	Last modified
<code>ModLabRole</code>	updates LabRole to allow it to assume itself	Zip	Python 3.9	4 days ago
<code>updateEntry</code>	-	Zip	Node.js 22.x	4 hours ago
<code>getEntries</code>	-	Zip	Node.js 22.x	6 hours ago
<code>createEntry</code>	-	Zip	Node.js 22.x	5 hours ago
<code>RedshiftEventSubscription</code>	Create Redshift event subscription to SNS Topic.	Zip	Python 3.9	4 days ago
<code>deleteEntry</code>	-	Zip	Node.js 22.x	5 hours ago
<code>MainMonitoringFunction</code>	-	Zip	Python 3.9	4 days ago
<code>RedshiftOverwatch</code>	Deletes Redshift Cluster if the count is more than 2.	Zip	Python 3.9	4 days ago
<code>RoleCreationFunction</code>	Create SLR if absent	Zip	Python 3.9	4 days ago

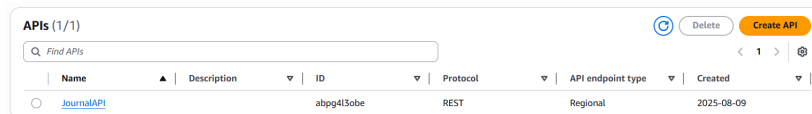
Фигура 6: Lambda в AWS Management Console

**Стъпка 4. Конфигуриране на API Gateway.**

**4.1. Създаване на Rest API.** Избрана е REST архитектура, тъй като тя е стандартен и доказан подход за изграждане на уеб API-та. Тя е stateless (безсъстоятелна), което я прави изключително мащабируема и лесна за интегриране с клиентски приложения, като осигурява ясен и предвидим начин за взаимодействие с ресурсите на бекенда чрез стандартни HTTP методи.

API name: JournalAPI,

Region: us-east



The screenshot shows the AWS API Gateway console with a list of 1 API. The API is named `JournalAPI` and is of type `REST`. The table has columns for Name, Description, ID, Protocol, API endpoint type, and Created.

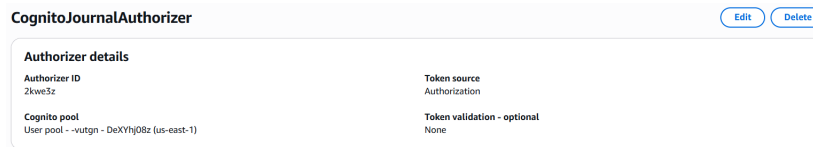
Name	Description	ID	Protocol	API endpoint type	Created
<code>JournalAPI</code>		<code>abpg4lSobe</code>	REST	Regional	2025-08-09

Фигура 7: APIs в AWS Management Console

**4.2. Създаване на Authorizer.** Авторизаторът е необходим, за да се осигури сигурност на API-то. Той интегрира API Gateway с Cognito User Pool, което позволява да се валидират потребителските токени (JWT) преди да се изпълни

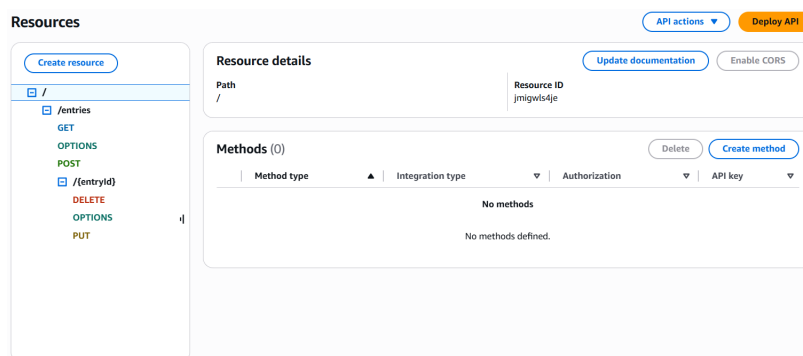
съответната Lambda функция. По този начин се гарантира, че само автентикирани потребители могат да достъпват и управляват своите дневникови записи, като се предотвратява нерегламентиран достъп.

Authorizer name: CognitoJournalAuthorizer Authorizer ID: 2kwe3z



Фигура 8: Authorizer в AWS Management Console

**4.3. Създаване на ресурси и методи и свързване с lambda функциите.** В API Gateway, ресурсите представляват пътищата (endpoints) на API-то, като например `/entries`. Методите са HTTP глаголите (GET, POST, PUT, DELETE), които се асоциират с тези ресурси. Всеки метод се интегрира с конкретна Lambda функция: `GET /entries` се свързва с `getEntries` Lambda функцията, `POST /entries` – с `createEntry`, `PUT /entries/{entryId}` – с `updateEntry`, `DELETE /entries/{entryId}` – с `deleteEntry`. Това позволява на клиентското приложение да изпраща заявки към специфични URL-и, които автоматично се пренасочват към правилната бекенд логика.



Фигура 9: JournalAPI Resources в AWS Management Console

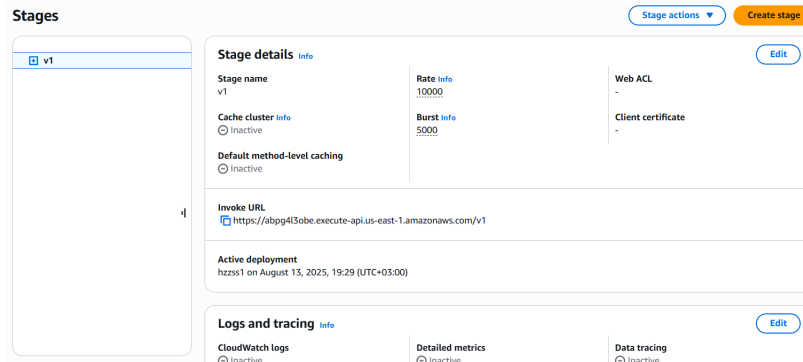
**4.4. Активиране на CORS.** Активирането на CORS (Cross-Origin Resource Sharing) е задължително, защото фронтенд приложението е хоствано на един домейн (S3 bucket), докато API Gateway се намира на друг. Без активиран CORS, уеб

браузърите ще блокират заявките от фронтенда към бекенда поради своята политика за сигурност, известна като Same-Origin Policy.

#### 4.5. Deploy на API.

Stage Name: v1,

Invoke URL: <https://abpg4l3obe.execute-api.us-east-1.amazonaws.com/v1>.

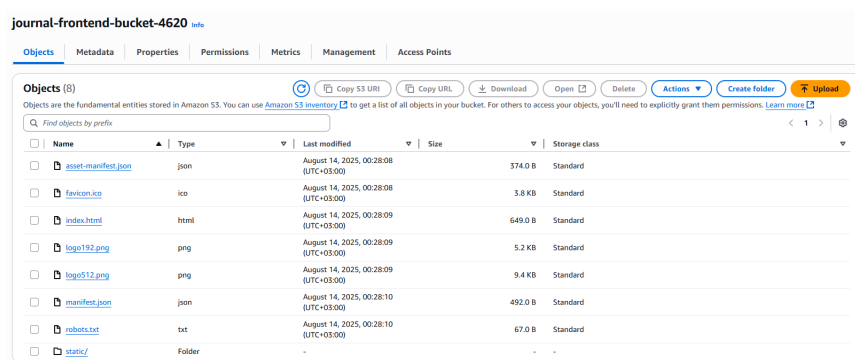


Фигура 10: JournalAPI Stages в AWS Management Console

#### Стъпка 5. Създаване на S3 Bucket за статичен уеб хостинг.

Bucket Name: journal-frontend-bucket-4620,

Bucket Website Endpoint: <http://journal-frontend-bucket-4620.s3-website-us-east-1.amazonaws.com>



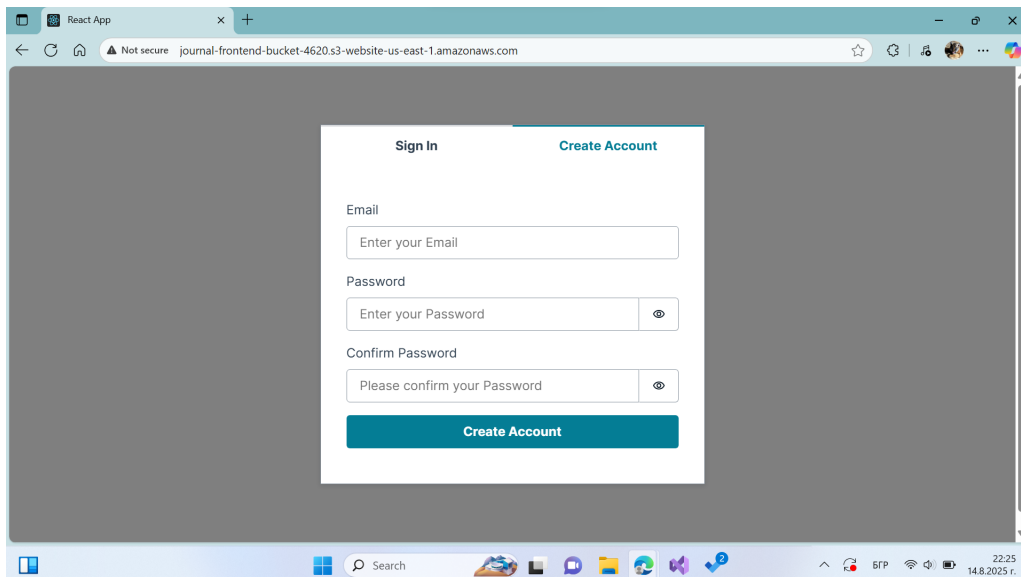
Фигура 11: S3 Bucket в AWS Management Console

## 6 Кратко ръководство за потребителя

Това ръководство ще ви запознае с интуитивния потребителски интерфейс на уеб дневника и как да използвате основните му функции. Приложението е разделено на няколко основни секции: автентикация, създаване на нов запис и преглед и управление на записи.

### 6.1 Регистрация и потвърждение на акаунт

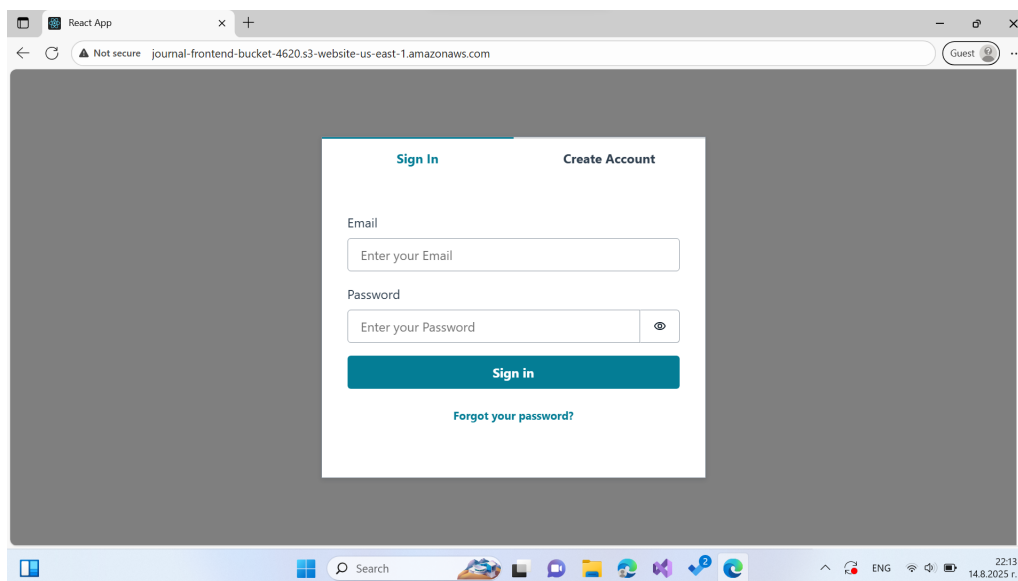
1. Отворете уеб приложението в браузъра си. В горната част на екрана ще видите опции за "Sign in" и "Create account".
2. Натиснете бутона **Create account**. Ще ви се отвори форма за регистрация.
3. Въведете валиден имейл адрес и изберете парола. Натиснете бутона **Create account**.
4. На посочения имейл адрес ще получите код за потвърждение (confirmation code). Въведете този код в полето **Confirmation code** и натиснете бутона **Confirm**.
5. Вашият акаунт вече е регистриран и потвърден.



Фигура 12: Екран за създаване на акаунт

## 6.2 Вход в системата

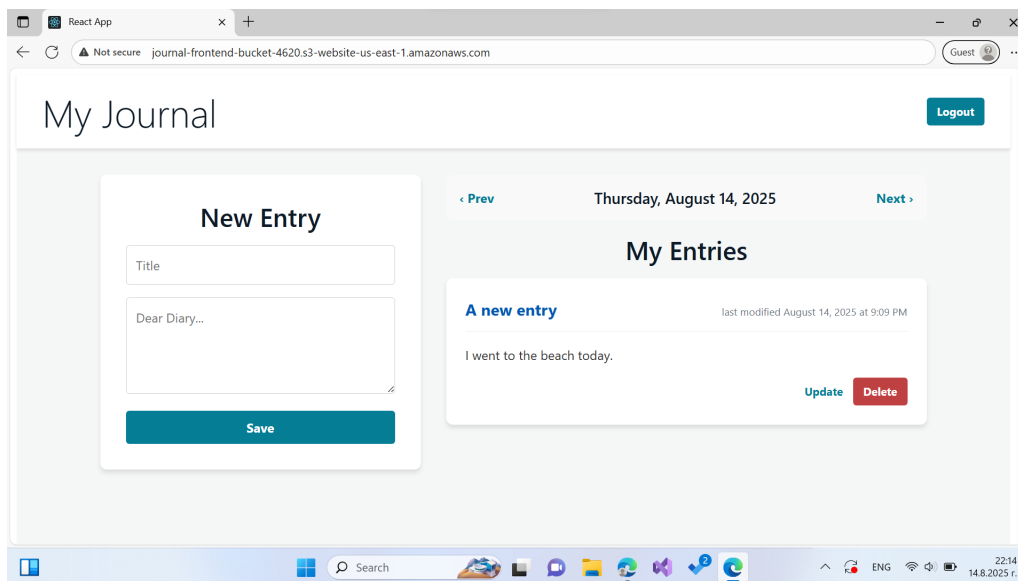
1. Отворете уеб приложението в браузъра си.
2. Въведете имейла и паролата, които сте използвали при регистрацията.
3. Натиснете бутона **Sign in**. Ще бъдете пренасочени към главната страница на дневника.



Фигура 13: Екран за вход в системата

## 6.3 Работа със записи в дневника

Работата със записи в дневника се извършва през основната страница.



Фигура 14: Основен екран

#### 1. Създаване на нов запис:

- На главната страница ще видите секция **New Entry**.
- Въведете заглавие на записа в полето **Title**.
- Въведете съдържанието на записа в текстовото поле **Dear diary....**
- Натиснете бутона **Save**. Новият запис ще се появи в секцията **My Entries**, сортирана по дата на създаване на записите.

#### 2. Преглед на записи:

- В секцията **My Entries** се показват всички ваши записи.
- Можете да навигирате между различните дати, като използвате бутоните **Prev** (предишен) и **Next** (следващ).

#### 3. Редактиране на запис:

- До всеки запис в списъка има бутони за управление.
- Натиснете бутона **Update**, за да промените заглавието или съдържанието на съществуващ запис.
- След като направите промените, натиснете отново **Update**, за да ги запазите.

#### 4. Изтриване на запис:

- За да изтриете запис, натиснете бутона **Delete** до съответния запис.
- Ще се появи изскачащ прозорец, който изисква потвърждение.
- Потвърдете, за да изтриете записа.

## 6.4 Изход от системата

За да прекратите сесията си, натиснете бутона **Logout**, който се намира в горния десен ъгъл на екрана.

## 7 Примерни данни

При създаване на запис потребителят задава заглавие и съдържание. Входните данни и допълнителна информация за времето на създаване и редактиране се записват от бекенда в `JournalEntries`.

Таблица 1: Примерни входни данни

	Title	Content	Date
1	A new entry	I went to the beach today.	2025-08-14
2	An old entry	I didn't go to the beach today.	2025-08-13

Таблица 2: Примерно съдържание на `JournalEntries`

	Запис 1	Запис 2
userId	34282458-2041-7085-9a84-29ae444e982f	34282458-2041-7085-9a84-29ae444e982f
entryId	3f0de891-f53d-4dd3-a797-62d9fb66b982	xqmf64j1-jjep-p2q1-uv78-pcq5dmporvyg
title	A new entry	An old entry
content	I went to the beach today.	I didn't go to the beach today.
createdAt	2025-08-14T18:09:17.631Z	2025-08-13T10:05:47.623Z
updatedAt	2025-08-14T18:09:17.631Z	2025-08-14T18:10:45.438Z

## 8 Описание на програмния код

Програмният код е разделен на няколко модула, които отговарят за различни аспекти на сървърлес архитектурата. Всеки модул е реализиран като AWS Lambda функция, обработваща конкретни API заявки.

## 8.1 Модул за автентикация (Cognito User Pool)

Този модул отговаря за управлението на потребителските идентичности. Той осигурява сигурна регистрация и вход в системата. Кодът на този модул включва логика за:

- Регистриране на нов потребител с имейл и парола.
- Изпращане на код за потвърждение по имейл.
- Потвърждаване на акаунт с получения код.
- Вход в системата и генериране на JWT (JSON Web Token) за автентикация.

Въпреки че самият код за тези операции е част от AWS Amplify или директно от SDK на AWS, логиката на приложението интегрира тези функционалности във фронтенд частта.

## 8.2 Модул за управление на записи (AWS Lambda функции)

Това е сърцето на бекенд логиката, реализирана чрез четири отделни Lambda функции. Всяка функция е проектирана да бъде отговорна само за една операция (single responsibility principle).

- `createEntry.js`: Тази функция се извиква при POST заявка към `/entries`. Тя получава данни за заглавие и съдържание, генерира уникални идентификатори (`entryId`, `createdAt`, `updatedAt`) и съхранява новия запис в DynamoDB таблицата.
- `getEntries.js`: Изпълнява се при GET заявка към `/entries`. Тя прави заявка към DynamoDB, използвайки `userId` от автентикирания потребител, за да върне всички дневникови записи за деня, сортирани по час на създаване.
- `editEntry.js`: Отговаря на PUT заявки към `/entries/{entryId}`. Функцията получава новите данни за запис и ги актуализира в DynamoDB, като предварително проверява дали потребителят, който прави заявката, е собственик на записа.
- `deleteEntry.js`: Тази функция се активира при DELETE заявка към `/entries/{entryId}`. Тя изтрива съответния запис от DynamoDB след проверка на правата на достъп на потребителя.



### 8.3 Модул на клиентската част (React приложение)

Клиентската част е разработена като Single Page Application (SPA) с React. Кодът включва:

- Компоненти за потребителския интерфейс (страница за вход, регистрация, основен екран на дневника).
- Бизнес логика за извикване на API ендпойнтите, обработка на отговорите и управление на състоянието на приложението.
- Интеграция с AWS Amplify, което улеснява взаимодействието с Cognito и API Gateway.

Цялото приложение е статично хоствано в S3 bucket.

## 9 Приноси на студента, ограничения и възможности за бъдещо развитие

В процеса на разработка е реализирано напълно функционално приложение за водене на дневник, което позволява създаване, преглед, редакция и изтриване на записи. Интерфейсът е интуитивен, а структурата на кода позволява лесно надграждане с нови модули и подобрения.

Сред основните ограничения на текущата версия е невъзможността за създаване на записи за минали дати. В настоящата реализация потребителят може да създава записи само за текущия ден.

Като възможност за бъдещо развитие се предлага:

- Добавяне на функционалност за създаване на записи за минали дни, което би позволило въвеждането на пропусната информация.
- Имплементиране на търсене по ключови думи и филтриране.
- Експорт и импорт на данни във формат PDF или CSV.

## 10 Какво научих

Основният принос от изпълнението на проекта беше придобиването на практически умения за работа с облачната платформа AWS. Научих се да интегрирам и конфигурирам услуги като *DynamoDB*, *Lambda*, *API Gateway*, *Cognito* и *S3*, за да изградя пълноценна бекенд инфраструктура. Паралелно с това развих умения за структуриране на софтуерен код, работа с  $\text{\LaTeX}$  за създаване на документация и прилагане на добри програмни практики.

## 11 Списък с фигури и таблици

### Списък на таблиците

1	Примерни входни данни . . . . .	15
2	Примерно съдържание на JournalEntries . . . . .	15

### Списък на фигурите

1	Архитектура на приложението в AWS . . . . .	4
2	JournalEntries Table в AWS Management Console . . . . .	7
3	JournalEntries Indexes в AWS Management Console . . . . .	7
4	User pool в AWS Management Console . . . . .	8
5	App client в AWS Management Console . . . . .	8
6	Lambda в AWS Management Console . . . . .	9
7	APIs в AWS Management Console . . . . .	9
8	Authorizer в AWS Management Console . . . . .	10
9	JournalAPI Resources в AWS Management Console . . . . .	10
10	JournalAPI Stages в AWS Management Console . . . . .	11
11	S3 Bucket в AWS Management Console . . . . .	11
12	Екран за създаване на акаунт . . . . .	12
13	Екран за вход в системата . . . . .	13
14	Основен екран . . . . .	14

## 12 Използвани източници

- [1] Amazon S3 Documentation
- [2] Amazon Cognito Documentation
- [3] Amazon API Gateway Documentation
- [4] AWS Lambda Documentation
- [5] Amazon DynamoDB Documentation