**Noor Ul Eman  
API Documentation**

**Zainab Eman | Imama Kainat | Noor Fatima**

**Base URL:** <http://localhost:5000/api>

**1. Register a New User**  
**Endpoint:** /auth/register  
**Method:** POST  
**Description:** Register a new user with name, email, and password.  
**Access:** Public

**Request Headers:**

* Content-Type: application/json

**Request Body:**  
{  
"name": "John Doe",  
"email": "john@example.com",  
"password": "mypassword123"  
}

**Success Response:**  
Status: 200 OK  
{  
"token": "<JWT Token>"  
}

**Error Response:**  
Status: 400 Bad Request  
{  
"errors": [  
{ "msg": "Email already in use" }  
]  
}

**2. Login User**  
**Endpoint:** /auth/login  
**Method:** POST  
**Description:** Authenticate user using email and password.  
**Access:** Public

**Request Headers:**

* Content-Type: application/json

**Request Body:**  
{  
"email": "john@example.com",  
"password": "mypassword123"  
}

**Success Response:**  
Status: 200 OK  
{  
"token": "<JWT Token>"  
}

**Error Response:**  
Status: 400 Bad Request  
{  
"errors": [  
{ "msg": "Invalid credentials" }  
]  
}

**3. Google OAuth Login**  
**Endpoint:** /auth/google  
**Method:** POST  
**Description:** Authenticate using a Google ID token.  
**Access:** Public

**Request Headers:**

* Content-Type: application/json

**Request Body:**  
{  
"token": "<Google ID Token>"  
}

**Success Response:**  
Status: 200 OK  
{  
"token": "<JWT Token>",  
"user": {  
"\_id": "user\_id",  
"name": "John Doe",  
"email": "john@example.com",  
"picture": "url",  
"isVerified": true,  
"createdAt": "timestamp",  
"updatedAt": "timestamp"  
}  
}

**Error Response:**  
Status: 400 or 401 or 500  
{  
"error": "Error message",  
"details": "Detailed message"  
}

**4. Get Current User**  
**Endpoint:** /auth/me  
**Method:** GET  
**Description:** Retrieve the currently authenticated user's information.  
**Access:** Private

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Success Response:**  
Status: 200 OK  
{  
"\_id": "user\_id",  
"name": "John Doe",  
"email": "john@example.com",  
"picture": "url",  
"isVerified": true,  
"createdAt": "timestamp",  
"updatedAt": "timestamp"  
}

**5. Forgot Password**  
**Endpoint:** /auth/forgot-password  
**Method:** POST  
**Description:** Send password reset link to user's email.  
**Access:** Public

**Request Body:**  
{  
"email": "john@example.com"  
}

**Success Response:**  
Status: 200 OK  
{  
"msg": "Reset link sent."  
}

**6. Verify Reset Token**  
**Endpoint:** /auth/reset-password/:token  
**Method:** GET  
**Description:** Verify if reset token is valid.  
**Access:** Public

**Success Response:**  
Status: 200 OK  
{  
"msg": "Token valid."  
}

**Error Response:**  
Status: 400 Bad Request  
{  
"msg": "Invalid or expired token."  
}

**7. Reset Password**  
**Endpoint:** /auth/reset-password/:token  
**Method:** POST  
**Description:** Submit a new password using a valid reset token.  
**Access:** Public

**Request Body:**  
{  
"password": "newpassword123"  
}

**Success Response:**  
Status: 200 OK  
{  
"msg": "Password updated."  
}

**8. Update Profile**  
**Endpoint:** /auth/profile  
**Method:** PUT  
**Description:** Update authenticated user’s profile (name and/or profile image).  
**Access:** Private

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Request Body:**  
{  
"name": "Updated Name",  
"profileImage": "image\_url"  
}

**Success Response:**  
Status: 200 OK  
{  
"\_id": "user\_id",  
"name": "Updated Name",  
"email": "john@example.com",  
"profileImage": "image\_url",  
"isVerified": true,  
"createdAt": "timestamp",  
"updatedAt": "timestamp"  
}

**9. Get Authenticated User’s Activity Logs**  
**Endpoint:** /auth/activity  
**Method:** GET  
**Description:** Return up to 100 most recent activity logs of the authenticated user.  
**Access:** Private

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Success Response:**  
Status: 200 OK  
[  
{  
"user": "user\_id",  
"action": "LOGIN",  
"details": {  
"email": "john@example.com"  
},  
"createdAt": "timestamp"  
}  
]

**10. Get Activity Logs (Direct Route)**  
**Endpoint:** /activity  
**Method:** GET  
**Description:** Fetches the last 100 formatted activity logs for the authenticated user.  
**Access:** Private

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Success Response:**  
Status: 200 OK  
[  
{  
"action": "REGISTER",  
"details": { "email": "john@example.com" },  
"timestamp": "2024-05-10T12:34:56.000Z"  
}  
]

**11. Get Adhkar by Type**  
**Endpoint:** /azkar  
**Method:** GET  
**Description:** Returns a list of adhkar strings filtered by category (e.g., morning, evening).  
**Access:** Private

**Query Parameters:**

* type: Optional. One of m, e, as, t, bs, wu, qd, pd

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Success Response:**  
Status: 200 OK  
{  
"category": "m",  
"count": 5,  
"items": [  
"Subhanallah",  
"Alhamdulillah",  
"Allahu Akbar"  
]  
}

**12. Get Morning Azkar**  
**Endpoint:** /azkar/morning  
**Method:** GET  
**Description:** Returns a list of morning azkar.  
**Access:** Private

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Success Response:**  
Status: 200 OK  
[  
{  
"text": "Some Morning Azkar",  
"reference": "Source"  
}  
]

**Error Response:**  
Status: 500 Internal Server Error  
{  
"msg": "Failed to load Morning Azkar"  
}

**13. Get Evening Azkar**  
**Endpoint:** /azkar/evening  
**Method:** GET  
**Description:** Returns a list of evening azkar.  
**Access:** Private

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Success Response:**  
Status: 200 OK  
[  
{  
"text": "Some Evening Azkar",  
"reference": "Source"  
}  
]

**Error Response:**  
Status: 500 Internal Server Error  
{  
"msg": "Failed to load Evening Azkar"  
}

**14. Get Post-Prayer Azkar**  
**Endpoint:** /azkar/post-prayer  
**Method:** GET  
**Description:** Returns a list of azkar to be recited after prayers.  
**Access:** Private

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Success Response:**  
Status: 200 OK  
[  
{  
"text": "Post Prayer Zikr",  
"reference": "Source"  
}  
]

**Error Response:**  
Status: 500 Internal Server Error  
{  
"msg": "Failed to load Post-Prayer Azkar"  
}

**15. Fetch Azkar from API (by type)**  
**Endpoint:** /azkar  
**Method:** GET  
**Description:** Fetch azkar or dua entries by type.  
**Access:** Private

**Query Parameters:**

* type: one of m, e, as, t, bs, wu, qd, pd
* json: true

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Success Response:**  
Status: 200 OK  
{  
"category": "m",  
"count": 5,  
"items": [  
"Zikr 1",  
"Zikr 2"  
]  
}

**Error Response:**  
Status: 500 Internal Server Error  
{  
"msg": "Failed to fetch Azkar/Dua"  
}

**16. Get All Bookmarks**  
**Endpoint:** /bookmarks  
**Method:** GET  
**Description:** Retrieve all bookmarks for the authenticated user.  
**Access:** Private

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Success Response:**  
Status: 200 OK  
[  
{  
"surahNumber": 1,  
"ayahNumber": 5,  
"category": "favourites",  
"notes": "My favorite verse"  
}  
]

**17. Get Bookmarks by Category**  
**Endpoint:** /bookmarks/category/:category  
**Method:** GET  
**Description:** Get bookmarks by specified category.  
**Access:** Private

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Success Response:**  
Status: 200 OK  
[  
{  
"surahNumber": 2,  
"ayahNumber": 255,  
"category": "knowledge",  
"notes": "Ayatul Kursi"  
}  
]

**18. Create a New Bookmark**  
**Endpoint:** /bookmarks  
**Method:** POST  
**Description:** Create a new bookmark entry.  
**Access:** Private

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Request Body:**  
{  
"surahNumber": 3,  
"ayahNumber": 190,  
"category": "reflection",  
"notes": "Very deep verse"  
}

**Success Response:**  
Status: 201 Created  
{  
"surahNumber": 3,  
"ayahNumber": 190,  
"category": "reflection",  
"notes": "Very deep verse"  
}

**19. Update a Bookmark**  
**Endpoint:** /bookmarks/:id  
**Method:** PUT  
**Description:** Update category or notes of a bookmark by ID.  
**Access:** Private

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Request Body:**  
{  
"category": "reflection",  
"notes": "Updated notes here"  
}

**Success Response:**  
Status: 200 OK  
{  
"surahNumber": 3,  
"ayahNumber": 190,  
"category": "reflection",  
"notes": "Updated notes here"  
}

**Error Response:**  
Status: 404 Not Found  
{  
"message": "Bookmark not found"  
}

**20. Delete a Bookmark**  
**Endpoint:** /bookmarks/:id  
**Method:** DELETE  
**Description:** Delete a specific bookmark by ID.  
**Access:** Private

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Success Response:**  
Status: 200 OK  
{  
"message": "Bookmark deleted successfully"  
}

**Error Response:**  
Status: 404 Not Found  
{  
"message": "Bookmark not found"  
}

**21. Export Bookmarks**  
**Endpoint:** /bookmarks/export  
**Method:** GET  
**Description:** Export all bookmarks as raw data without user metadata.  
**Access:** Private

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Success Response:**  
Status: 200 OK  
[  
{  
"surahNumber": 1,  
"ayahNumber": 1,  
"category": "memorize",  
"notes": "Start of Quran"  
}  
]

**22. Import Bookmarks**  
**Endpoint:** /bookmarks/import  
**Method:** POST  
**Description:** Bulk import bookmarks.  
**Access:** Private

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Request Body:**  
[  
{  
"surahNumber": 1,  
"ayahNumber": 2,  
"category": "memorize",  
"notes": "Second verse"  
}  
]

**Success Response:**  
Status: 200 OK  
{  
"message": "Bookmarks imported successfully"  
}

**Error Response:**  
Status: 400 Bad Request  
{  
"message": "Failed to import bookmarks"  
}

**23. Get Today’s Calendar (Prayer Times)**  
**Endpoint:** /calendar/today  
**Method:** GET  
**Description:** Returns today’s Islamic calendar data, including prayer times.  
**Access:** Public (no authentication required)

**Success Response:**  
Status: 200 OK  
{  
"fajr": "04:12",  
"dhuhr": "12:08",  
"asr": "15:37",  
"maghrib": "18:52",  
"isha": "20:15"  
}

**Error Responses:**  
Status: 400 Bad Request  
{  
"msg": "Prayer settings not found. Please configure your prayer settings first.",  
"code": "PRAYER\_SETTINGS\_MISSING"  
}

Status: 503 Service Unavailable  
{  
"msg": "External calendar service is currently unavailable",  
"code": "EXTERNAL\_API\_ERROR"  
}

Status: 500 Internal Server Error  
{  
"msg": "An unexpected error occurred",  
"code": "INTERNAL\_ERROR",  
"details": "Error details"  
}

**24. Get Chat History**  
**Endpoint:** /chat/history  
**Method:** GET  
**Description:** Retrieve community chat history. Creates a new chat thread if one doesn’t exist.  
**Access:** Private

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Success Response:**  
Status: 200 OK  
[  
{  
"sender": { "name": "User", "picture": "url" },  
"text": "Message content",  
"timestamp": "2024-05-11T12:00:00Z"  
}  
]

**25. Get Single Chat Message**  
**Endpoint:** /chat/message/:messageId  
**Method:** GET  
**Description:** Retrieve a single message by its ID.  
**Access:** Private

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Success Response:**  
Status: 200 OK  
{  
"sender": "user\_id",  
"text": "Message text",  
"timestamp": "2024-05-11T12:00:00Z"  
}

**Error Response:**  
Status: 404 Not Found  
{ "message": "Message not found" }

**26. Delete a Chat Message**  
**Endpoint:** /chat/message/:messageId  
**Method:** DELETE  
**Description:** Delete a message. Only sender or admin can delete.  
**Access:** Private

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Success Response:**  
Status: 200 OK  
{ "message": "Message deleted successfully" }

**Error Response:**  
Status: 403 Forbidden  
{ "message": "Not authorized to delete this message" }

Status: 404 Not Found  
{ "message": "Message not found" }

**27. Mark Messages as Read**  
**Endpoint:** /chat/read  
**Method:** POST  
**Description:** Mark an array of messages as read.  
**Access:** Private

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Request Body:**  
{  
"messageIds": ["msgid1", "msgid2"]  
}

**Success Response:**  
Status: 200 OK  
{ "message": "Messages marked as read" }

**28. Get Random Hadith / Dua**  
**Endpoint:** /dua/hadith  
**Method:** GET  
**Description:** Fetch a random hadith or dua from the external API.  
**Access:** Private

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Success Response:**  
Status: 200 OK  
{  
"text": "Random Hadith or Dua",  
"reference": "Book, Volume, Number"  
}

**Error Response:**  
Status: 500 Internal Server Error  
{ "msg": "Could not fetch hadith" }

**29. Get All Events**  
**Endpoint:** /events  
**Method:** GET  
**Description:** Retrieve all user-created events sorted by Hijri date.  
**Access:** Private

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Success Response:**  
Status: 200 OK  
[  
{  
"title": "Eid",  
"hijriDate": "10-10-1445",  
"gregDate": "2024-04-10"  
}  
]

**30. Create an Event**  
**Endpoint:** /events  
**Method:** POST  
**Description:** Add a new event with Hijri and Gregorian dates.  
**Access:** Private

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Request Body:**  
{  
"title": "Laylatul Qadr",  
"hijriDate": "27-09-1445",  
"gregDate": "2024-04-07"  
}

**Success Response:**  
Status: 200 OK  
{  
"title": "Laylatul Qadr",  
"hijriDate": "27-09-1445",  
"gregDate": "2024-04-07"  
}

**Error Response:**  
Status: 400 Bad Request  
{ "msg": "You already have that event on this date." }

**31. Delete an Event**  
**Endpoint:** /events/:id  
**Method:** DELETE  
**Description:** Delete an event by its ID.  
**Access:** Private

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Success Response:**  
Status: 200 OK  
{ "success": true }

**32. Get All Hadith Books**  
**Endpoint:** /library/books  
**Method:** GET  
**Description:** Retrieve a list of all supported hadith books (editions).  
**Access:** Private

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Success Response:**  
Status: 200 OK  
[ "bukhari", "muslim", "tirmidhi" ]

**33. Get Chapters of a Book**  
**Endpoint:** /library/books/:edition/chapters  
**Method:** GET  
**Description:** Retrieve all chapters for the given book edition.  
**Access:** Private

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Success Response:**  
Status: 200 OK  
[ "Chapter 1", "Chapter 2" ]

**34. Get Hadiths by Chapter**  
**Endpoint:** /library/books/:edition/chapters/:chapter  
**Method:** GET  
**Description:** Get all hadiths for a specific chapter of a book.  
**Access:** Private

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Success Response:**  
Status: 200 OK  
[  
{  
"number": 1,  
"text": "The Prophet said...",  
"grade": "Sahih"  
}  
]

**35. Search Hadiths in Chapter**  
**Endpoint:** /library/books/:edition/chapters/:chapter/search  
**Method:** GET  
**Description:** Search within a specific chapter for a query.  
**Access:** Private

**Query Parameter:**

* q: search term

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Success Response:**  
Status: 200 OK  
[  
{  
"number": 1,  
"text": "Search result content"  
}  
]

**36. Add Hadith to Favorites**  
**Endpoint:** /library/favorites  
**Method:** POST  
**Description:** Add a hadith to user’s favorites.  
**Access:** Private

**Request Body:**  
{  
"edition": "bukhari",  
"chapter": "1",  
"number": 5,  
"hadith": "Text of the hadith"  
}

**Success Response:**  
Status: 200 OK  
{ "number": 5, "hadith": "Text of the hadith" }

**Error Response:**  
Status: 400 Bad Request  
{ "msg": "Already favorited" }

**37. Remove Hadith from Favorites**  
**Endpoint:** /library/favorites/:id  
**Method:** DELETE  
**Description:** Remove a hadith from user’s favorites by ID.  
**Access:** Private

**Success Response:**  
Status: 200 OK  
{ "success": true }

**38. Get Prayer Times**  
**Endpoint:** /prayer/times  
**Method:** POST  
**Description:** Retrieve today's prayer times for the given coordinates.  
**Access:** Private

**Request Body:**  
{  
"latitude": 31.5204,  
"longitude": 74.3587,  
"method": 2  
}

**Success Response:**  
Status: 200 OK  
{  
"fajr": "04:00",  
"dhuhr": "12:30",  
"asr": "15:45",  
"maghrib": "18:55",  
"isha": "20:10",  
"completed": [ "fajr", "dhuhr" ]  
}

**39. Toggle Prayer Log**  
**Endpoint:** /prayer/log  
**Method:** POST  
**Description:** Log or un-log a prayer for a specific day.  
**Access:** Private

**Request Body:**  
{  
"prayer": "maghrib",  
"date": "2024-05-11"  
}

**Success Response:**  
Status: 200 OK  
{ "completed": [ "fajr", "maghrib" ] }

**Error Response:**  
Status: 400 Bad Request  
{ "msg": "Invalid prayer name" }

**40. Get Prayer Stats**  
**Endpoint:** /prayer/stats  
**Method:** GET  
**Description:** Get daily prayer completion stats for the last X days.  
**Access:** Private

**Query Parameter:**

* days (optional): number of days to fetch stats for (default is 7)

**Success Response:**  
Status: 200 OK  
[  
{  
"date": "2024-05-10",  
"completed": 3,  
"details": [  
{ "name": "fajr", "completed": true },  
{ "name": "dhuhr", "completed": false },  
{ "name": "asr", "completed": true },  
{ "name": "maghrib", "completed": true },  
{ "name": "isha", "completed": false }  
]  
}  
]

**41. Get Quran Bookmark (Progress)**  
**Endpoint:** /quran/progress  
**Method:** GET  
**Description:** Get the user’s current bookmark (surah and ayah).  
**Access:** Private

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Success Response:**  
Status: 200 OK  
{  
"surah": 36,  
"ayah": 12,  
"updatedAt": "2024-05-11T18:30:00Z"  
}

**42. Set or Update Quran Bookmark**  
**Endpoint:** /quran/progress  
**Method:** POST  
**Description:** Save or update the user’s bookmark.  
**Access:** Private

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Request Body:**  
{  
"surah": 36,  
"ayah": 12  
}

**Success Response:**  
Status: 200 OK  
{  
"surah": 36,  
"ayah": 12,  
"updatedAt": "2024-05-11T18:30:00Z"  
}

**43. Get Full Surah**  
**Endpoint:** /quran/surah/:num  
**Method:** GET  
**Description:** Retrieve full surah text and translation by number.  
**Access:** Private

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Success Response:**  
Status: 200 OK  
{  
"name": "Yasin",  
"verses": [ { "ayah": 1, "text": "...", "translation": "..." }, ... ]  
}

**Error Response:**  
Status: 500 Internal Server Error  
{ "msg": "Failed to fetch surah" }

**44. Get Quran Audio URL**  
**Endpoint:** /quran/audio/:surah/:ayah  
**Method:** GET  
**Description:** Returns the URL to stream audio for the given ayah.  
**Access:** Private

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Success Response:**  
Status: 200 OK  
{ "url": "https://cdn.quran.com/audio/..." }

**45. Get All Ramadan Goals**  
**Endpoint:** /ramadan-goals  
**Method:** GET  
**Description:** Retrieve all Ramadan goals for the authenticated user.  
**Access:** Private

**Request Headers:**

* Authorization: Bearer <JWT Token>

**Success Response:**  
Status: 200 OK  
[  
{  
"title": "Pray Tahajjud",  
"type": "spiritual",  
"date": "2024-04-01"  
}  
]

**46. Create a Ramadan Goal**  
**Endpoint:** /ramadan-goals  
**Method:** POST  
**Description:** Create a new Ramadan goal.  
**Access:** Private

**Request Body:**  
{  
"title": "Give Sadaqah",  
"description": "Donate daily",  
"type": "charity",  
"date": "2024-04-03"  
}

**Success Response:**  
Status: 201 Created  
{ "title": "Give Sadaqah", "type": "charity" }

**47. Update a Ramadan Goal**  
**Endpoint:** /ramadan-goals/:id  
**Method:** PATCH  
**Description:** Update an existing goal.  
**Access:** Private

**Request Body:**  
{ "description": "Updated note" }

**Success Response:**  
Status: 200 OK  
{ "description": "Updated note" }

**Error Response:**  
Status: 404 Not Found  
{ "message": "Goal not found" }

**48. Delete a Ramadan Goal**  
**Endpoint:** /ramadan-goals/:id  
**Method:** DELETE  
**Description:** Delete a specific goal by ID.  
**Access:** Private

**Success Response:**  
Status: 200 OK  
{ "message": "Goal deleted" }

**49. Get Ramadan Goals by Date Range**  
**Endpoint:** /ramadan-goals/range  
**Method:** GET  
**Description:** Get all goals within a specified date range.  
**Access:** Private

**Query Parameters:**

* startDate
* endDate

**Success Response:**  
Status: 200 OK  
[  
{ "title": "Read Surah Mulk", "date": "2024-04-10" }  
]

**50. Get All Reflections**  
**Endpoint:** /reflections  
**Method:** GET  
**Description:** Get all reflections for the authenticated user.  
**Access:** Private

**Success Response:**  
Status: 200 OK  
[  
{  
"content": "Today I felt grateful.",  
"mood": "grateful",  
"date": "2024-05-11"  
}  
]

**51. Create a Reflection**  
**Endpoint:** /reflections  
**Method:** POST  
**Description:** Create a new reflection.  
**Access:** Private

**Request Body:**  
{  
"content": "Reflected on Surah Kahf",  
"mood": "peaceful",  
"tags": ["Quran", "Friday"],  
"date": "2024-05-10"  
}

**Success Response:**  
Status: 201 Created  
{ "content": "Reflected on Surah Kahf" }

**52. Update a Reflection**  
**Endpoint:** /reflections/:id  
**Method:** PATCH  
**Description:** Update an existing reflection.  
**Access:** Private

**Request Body:**  
{ "mood": "inspired" }

**Success Response:**  
Status: 200 OK  
{ "mood": "inspired" }

**Error Response:**  
Status: 404 Not Found  
{ "message": "Reflection not found" }

**53. Delete a Reflection**  
**Endpoint:** /reflections/:id  
**Method:** DELETE  
**Description:** Delete a specific reflection.  
**Access:** Private

**Success Response:**  
Status: 200 OK  
{ "message": "Reflection deleted" }

**54. Get Reflections by Date Range**  
**Endpoint:** /reflections/range  
**Method:** GET  
**Description:** Retrieve reflections within a specified date range.  
**Access:** Private

**Query Parameters:**

* startDate
* endDate

**Success Response:**  
Status: 200 OK  
[  
{ "content": "Day 1 Ramadan reflections" }  
]

**55. Get Reflections by Mood**  
**Endpoint:** /reflections/mood/:mood  
**Method:** GET  
**Description:** Filter reflections based on mood type.  
**Access:** Private

**Success Response:**  
Status: 200 OK  
[  
{ "content": "I felt peaceful after prayer." }  
]

**56. Log a Prayer (Salah)**  
**Endpoint:** /salah/log  
**Method:** POST  
**Description:** Log a specific prayer as completed for a date.  
**Access:** Private

**Request Body:**  
{  
"prayer": "fajr",  
"date": "2024-05-10"  
}

**Success Response:**  
Status: 200 OK  
{  
"prayer": "fajr",  
"date": "2024-05-10"  
}

**Error Response:**  
Status: 400 Bad Request  
{ "msg": "Prayer already logged" }

**57. Remove Logged Prayer (Uncheck)**  
**Endpoint:** /salah/log  
**Method:** DELETE  
**Description:** Remove a logged prayer.  
**Access:** Private

**Query Parameters or Body:**

* prayer
* date

**Success Response:**  
Status: 200 OK  
{ "msg": "Prayer log removed" }

**Error Response:**  
Status: 404 Not Found  
{ "msg": "Prayer log not found" }

**58. Get Prayer Logs (Date Range)**  
**Endpoint:** /salah/logs  
**Method:** GET  
**Description:** Retrieve all logged prayers within a date range.  
**Access:** Private

**Query Parameters:**

* start: YYYY-MM-DD
* end: YYYY-MM-DD

**Success Response:**  
Status: 200 OK  
[  
{ "prayer": "fajr", "date": "2024-05-10" },  
{ "prayer": "isha", "date": "2024-05-10" }  
]

**59. Get Salah Stats**  
**Endpoint:** /salah/stats  
**Method:** GET  
**Description:** Get prayer completion stats for a number of past days.  
**Access:** Private

**Query Parameters:**

* days (optional, default: 30)

**Success Response:**  
Status: 200 OK  
[  
{  
"date": "2024-05-10",  
"completed": 4,  
"prayers": ["fajr", "dhuhr", "asr", "isha"],  
"details": [  
{ "name": "fajr", "completed": true },  
{ "name": "maghrib", "completed": false }  
]  
}  
]

**60. Get Quran Editions**  
**Endpoint:** /search/editions  
**Method:** GET  
**Description:** Retrieve all available Quran editions.  
**Access:** Public

**Success Response:**  
Status: 200 OK  
[ "en.sahih", "ar.qurancomplex" ]

**61. Get Quran Languages**  
**Endpoint:** /search/languages  
**Method:** GET  
**Description:** Retrieve list of supported languages.  
**Access:** Public

**Success Response:**  
Status: 200 OK  
[ "en", "ar", "ur" ]

**62. Get Quran Types**  
**Endpoint:** /search/types  
**Method:** GET  
**Description:** Get the list of available Quran types (e.g., tafsir, translation).  
**Access:** Public

**Success Response:**  
Status: 200 OK  
[ "translation", "tafsir" ]

**63. Get Quran Formats**  
**Endpoint:** /search/formats  
**Method:** GET  
**Description:** Get the list of available Quran formats.  
**Access:** Public

**Success Response:**  
Status: 200 OK  
[ "text", "audio", "image" ]

**64. Get Juz List**  
**Endpoint:** /search/juz  
**Method:** GET  
**Description:** Retrieve list of 30 Juz with basic metadata.  
**Access:** Public

**Success Response:**  
Status: 200 OK  
[ { "number": 1, "name": "Juz 1", "startSurah": 1, "endSurah": 2 } ]

**65. Get Ruku List for Surah**  
**Endpoint:** /search/ruku/:surahNumber  
**Method:** GET  
**Description:** Get rukus in a given surah.  
**Access:** Public

**Success Response:**  
Status: 200 OK  
[ { "number": 1, "startAyah": 1, "endAyah": 10 } ]

**66. Get Revelation Types**  
**Endpoint:** /search/revelation-types  
**Method:** GET  
**Description:** Returns fixed revelation types: Meccan, Medinan.  
**Access:** Public

**Success Response:**  
Status: 200 OK  
[ "Meccan", "Medinan" ]

**67. Search Quran**  
**Endpoint:** /search/search  
**Method:** GET  
**Description:** Search the Quran by text, surah, and edition.  
**Access:** Public

**Query Parameters:**

* q: search term
* surah (default: all)
* edition (default: en)

**Success Response:**  
Status: 200 OK  
[ { "text": "..." } ]

**68. Get Search Suggestions**  
**Endpoint:** /search/suggestions  
**Method:** GET  
**Description:** Returns suggested results based on the search query.  
**Access:** Private

**Query Parameter:**

* q: search term

**Success Response:**  
Status: 200 OK  
[ { "text": "Alhamdulillah", "translation": "All praise is due to Allah" } ]

**69. Get Tasbih Counters**  
**Endpoint:** /tasbih  
**Method:** GET  
**Description:** Get all tasbih counters for a user.  
**Access:** Private

**Success Response:**  
Status: 200 OK  
[ { "name": "SubhanAllah", "count": 33, "category": "Morning" } ]

**70. Get Tasbih Categories**  
**Endpoint:** /tasbih/categories  
**Method:** GET  
**Description:** Returns list of tasbih categories (default + user-defined).  
**Access:** Private

**Success Response:**  
Status: 200 OK  
[ "Morning", "Evening", "Custom" ]

**71. Create a Tasbih Counter**  
**Endpoint:** /tasbih  
**Method:** POST  
**Description:** Create a new tasbih counter.  
**Access:** Private

**Request Body:**  
{  
"name": "SubhanAllah",  
"target": 33,  
"description": "Morning dhikr",  
"category": "Morning"  
}

**Success Response:**  
Status: 200 OK  
{ "name": "SubhanAllah", "target": 33 }

**72. Increment a Counter**  
**Endpoint:** /tasbih/:id/inc  
**Method:** PATCH  
**Description:** Increment tasbih counter and update streak.  
**Access:** Private

**Success Response:**  
Status: 200 OK  
{ "count": 34, "streak": 2 }

**73. Decrement a Counter**  
**Endpoint:** /tasbih/:id/dec  
**Method:** PATCH  
**Description:** Decrement tasbih counter.  
**Access:** Private

**Request Body (optional):**  
{ "step": 1 }

**Success Response:**  
Status: 200 OK  
{ "count": 32 }

**74. Reset a Counter**  
**Endpoint:** /tasbih/:id/reset  
**Method:** PATCH  
**Description:** Reset count and streak to 0.  
**Access:** Private

**Success Response:**  
Status: 200 OK  
{ "count": 0, "streak": 0 }

**75. Update Counter Details**  
**Endpoint:** /tasbih/:id  
**Method:** PATCH  
**Description:** Edit counter metadata (name, target, etc).  
**Access:** Private

**Request Body:**  
{ "name": "New Name", "target": 100 }

**Success Response:**  
Status: 200 OK  
{ "name": "New Name", "target": 100 }

**76. Delete Counter**  
**Endpoint:** /tasbih/:id  
**Method:** DELETE  
**Description:** Delete a tasbih counter.  
**Access:** Private

**Success Response:**  
Status: 200 OK  
{ "success": true }