**🌀** **🌀 BbSp – Blockchain-backed Synchronization Protocol**

**BbSp הוא פרוטוקול סנכרון מאובטח מבוסס P2P שמורכב משתי שכבות מתואמות:**

**1. Application Sync Layer (שכבת תיאום אפליקטיבית)**

**מטרה: סנכרון סטטוסים בין שני מכשירים (peer ↔ peer).**

**רץ כל 4 שניות (syncWithPeer)**

**שולח את רשימת הסטטוסים הידועים של ההודעות ל־peer.**

**מקבל עדכוני סטטוס מה-peer ומעדכן מקומית כל הודעה (כולל חתימת hash מחדש).**

**שינויים נרשמים ב־syncLogs לצורך תיעוד ובקרה.**

**תומך ב־forceSync() לסנכרון כפוי לפי דרישת TRUST.**

**📡 נתיב קריטי: שליחה חוזרת של הודעות שלא התקבלו**

**אם קיימות הודעות בסטטוס `not\_delivered` עם `receiverId`, מתבצעת שליחה חוזרת מיידית בעת כל `syncWithPeer` או `forceSync`. אין אפשרות להשלמת בלוק חדש (TRUST Block) אם קיימות הודעות שלא נשלחו ליעדן. הודעות אלה נשלחות ל־peer אם הוא מחובר, ורק לאחר שליחה זו יבוצע סנכרון סטטוסים מלא.**

**🔄 Sync Status UI:**

**🔴 לא מסונכרן (אין peer / כשל)**

**🟡 בתהליך סנכרון**

**🟢 סנכרון מלא**

**2. TRUST Ledger Layer (שכבת אימות בלוקצ'יין)**

**מטרה: לאמת את תקינות כל ההודעות והסטטוסים בשרשרת בלתי ניתנת לזיוף.**

**רץ כל 5 שניות (syncLedgers)**

**יוצר Ledger (רשימת id + status + hash).**

**מייצר בלוק חדש עם createBlock (כולל previousHash).**

**שולח ל-peer להשוואה:**

**אם יש הודעות חסרות – נמשכות מה-peer (requestMissingMessage)**

**אם יש סטטוסים/hash שונה – מתבצע עדכון**

**אם יש אי התאמה – מופעל forceSync() מהשכבה האפליקטיבית.**

**בלוק נשמר רק לאחר התאמה מלאה בין שני הצדדים.**

**נורית סנכרון משתנה בהתאם להתאמת ה־Blockchain.**

**🧾 עקרונות אבטחה ותיעוד**

**כל הודעה נחתמת ב־ SHA256 לפי - id|status|text|audioBase64|imageBase64|videoBase64**

**כל בלוק נושא hash מלא + hash קודם (לכפיית סדר כרונולוגי)**

**Blockchain נשמר ב־AsyncStorage (trustBlocks)**

**כל פעולת sync נרשמת בלוג (syncLogs) כולל סיבת sync (רגיל / mismatch)**

**✅ בקרת תאימות לפרויקטים אחרים**

**כל מערכת שתומכת ב־BbSp חייבת לעמוד בתנאים הבאים:**

**סנכרון סטטוסים עם hash חתום מחדש בכל עדכון.**

**אימות Ledger דו-צדדי ויצירת בלוק חדש רק לאחר התאמה.**

**הפעלה של forceSync במקרה של mismatch.**

**שליחה חוזרת של כל הודעה שלא התקבלה ליעדה.**

**שמירת כל היסטוריית הבלוקים באופן מקומי בלבד.**

**שקיפות מלאה בנורית הסנכרון למשתמש. – Blockchain-backed Synchronization Protocol**

**BbSp** הוא פרוטוקול סנכרון מאובטח מבוסס P2P שמורכב משתי שכבות מתואמות:

**1. Application Sync Layer (שכבת תיאום אפליקטיבית)**

**מטרה:** סנכרון סטטוסים בין שני מכשירים (peer ↔ peer).

* רץ כל 4 שניות (syncWithPeer)
* שולח את רשימת הסטטוסים הידועים של ההודעות ל־peer.
* מקבל עדכוני סטטוס מה-peer ומעדכן מקומית כל הודעה (כולל חתימת hash מחדש).
* שינויים נרשמים ב־syncLogs לצורך תיעוד ובקרה.
* תומך ב־forceSync() לסנכרון כפוי לפי דרישת TRUST.

🔄 Sync Status UI:

* 🔴 לא מסונכרן (אין peer / כשל)
* 🟡 בתהליך סנכרון
* 🟢 סנכרון מלא

**2. TRUST Ledger Layer (שכבת אימות בלוקצ'יין)**

**מטרה:** לאמת את תקינות כל ההודעות והסטטוסים בשרשרת בלתי ניתנת לזיוף.

* רץ כל 5 שניות (syncLedgers)
* יוצר Ledger (רשימת id + status + hash).
* מייצר בלוק חדש עם createBlock (כולל previousHash).
* שולח ל-peer להשוואה:
  + אם יש הודעות חסרות – נמשכות מה-peer (requestMissingMessage)
  + אם יש סטטוסים/hash שונה – מתבצע עדכון
* אם יש אי התאמה – מופעל forceSync() מהשכבה האפליקטיבית.
* בלוק נשמר רק לאחר התאמה מלאה בין שני הצדדים.
* נורית סנכרון משתנה בהתאם להתאמת ה־Blockchain.

**🧾 עקרונות אבטחה ותיעוד**

* כל הודעה נחתמת ב־ SHA256 לפי - id|status|text|audioBase64|imageBase64|videoBase64
* כל בלוק נושא hash מלא + hash קודם (לכפיית סדר כרונולוגי)
* Blockchain נשמר ב־AsyncStorage (trustBlocks)
* כל פעולת sync נרשמת בלוג (syncLogs) כולל סיבת sync (רגיל / mismatch)

**✅ בקרת תאימות לפרויקטים אחרים**

כל מערכת שתומכת ב־BbSp חייבת לעמוד בתנאים הבאים:

1. **סנכרון סטטוסים** עם hash חתום מחדש בכל עדכון.
2. **אימות Ledger** דו-צדדי ויצירת בלוק חדש רק לאחר התאמה.
3. **הפעלה של forceSync** במקרה של mismatch.
4. **שמירת כל היסטוריית הבלוקים באופן מקומי בלבד.**
5. **שקיפות מלאה בנורית הסנכרון למשתמש.**