רשת חברתית – פייקלוק – שלב ב'

בשלב הראשון של הפרוייקט פיתחתם רשת חברתית מעל טכנולוגיות Angular, Node.js ו-SQL.
בשלב השני של הפרוייקט תמשיכו לפתח את הרשת החברתית ותשלבו בה טכנולוגיות נוספות.
המשימות העיקריות שלכם בשלב השני יהיו:

- יצירת dockers
- הוספת ElasticSearch ו-Kibana
- הוספת swagger
- שינוי קוד צד השרת כך שיעבוד עם dependency injection
- הוספת unit tests
- הוספת logs
- הוספת תקשורת דו כיוונית בין השרת ל-client

על מנת להתחיל לבצע את המשימות של השלב השני של הפרוייקט חובה לקבל אישור מרועי

יצירת dockers

בשלב הראשון של הפרוייקט כתבתם את קוד השרת ב-Node.js.
בשלב השני של הפרוייקט תיצרו docker שיריץ את הקוד של צד השרת.
בשלב הראשון של הפרוייקט כתבתם את קוד ה-client ב-Angular.
בשלב השני של הפרוייקט תיצרו docker שיריץ את הקוד של צד ה-client.

הוספת ElasticSearch

בשלב הראשון של הפרוייקט ה-DB היחיד אותו פיתחתם היה SQL.
בשלב השני של הפרוייקט תעבירו חלק מהמידע ששמרתם ב-SQL ל-ElasticSearch.
כל החלק של זיהוי משתמשים יישאר ב-SQL.
כל החלקים האחרים יעברו ל-ElasticSearch.
על מנת לעבוד עם ElasticSearch יהיה עליכם להקים docker.
יהיה עליכם לשנות את הקוד ב-Node.js כך שיעבוד עם ElasticSearch במקום עם SQL.

הוספת Kibana

בשלב השני של הפרוייקט תתחילו לעבוד עם Kibana.
בשלב השני של הפרוייקט תעבדו רק עם חלק ה-dev tools של Kibana בשביל לתשאל את ElasticSearch.
בשלב השלישי של הפרוייקט תעבדו עם דברים נוספים של kibana.
על מנת לעבוד עם Kibana יהיה עליכם להקים docker.

הוספת Swagger

בשלב הראשון של הפרוייקט כתבתם בצד השרת service שחושף שירותים שונים.
אולם מי שמעוניין לקרוא לשירותים אלו לא מודע לכך שהשירותים האלו קיימים.
על מנת לחשוף אילו שירותים קיימים בשרת נהוג להשתמש ב-swagger.
בשלב השני של הפרוייקט יהיה עליכם להוסיף swagger.
ניתן להשתמש (לא חובה) ב-swagger tools.

שינוי קוד צד השרת כך שיעבוד עם dependency injection

בשלב הראשון של הפרוייקט צד ה-client שנכתב ב-angular היה מבוסס על dependency injection (לדוגמא זה האופן שבו components מקבלים את ה-services).
מטרתנו בשלב השני של הפרוייקט לפתח גם את צד השרת באמצעות dependency injection.
לשם כך ניתן להשתמש ב-Kontainer-di (לא חובה).
עבודה נכונה עם dependency injection מאד מקלה עלינו כאשר אנו מעוניינים להוסיף unit tests.

הוספת unit tests

עליך להוסיף unit tests לצד השרת (Node.js). על ה-unit tests להיות כתובים ב-mocha, chai ו-sinon.
עליך לכתוב unit tests לצד ה-client (angular) אותו פיתחת בשלב הראשון של הפרוייקט.
השתמש ב-mocks במידת הצורך.

הוספת logs

עליך להוסיף logs לצד השרת של הפרוייקט.
לדוגמא ניתן להשתמש ב-Winston (לא חובה).
עליך לאפשר גישה ל-logs מחוץ ל-docker כך שה-logs לא יימחקו ביחד עם ה-docker (persistent).
ניתן להשתמש לצורך כך ב-volume (לא חובה).

הוספת תקשורת דו כיוונית בין השרת ל-client

בשלב הראשון של הפרוייקט לא היתה תקשורת דו כיוונית בין ה-client לשרת.
לדוגמא, אם שני משתמשים פתחו browser ואחד עשה like ל-post, השני לא היה רואה את ה-like שביצע המשתמש הראשון אלא אם היה מבצע refresh לעמוד.
בחלק השני של הפרוייקט תוסיף תקשורת דו כיוונית בין ה-client לשרת.
תבצע זאת באמצעות socket-io.