Soluzione

1. (A) [X] (B) [ ] (C) [ ] (D) [ ]
2. (A) [ ] (B) [ ] (C) [ ] (D) [X]
3. (A) [ ] (B) [X] (C) [ ] (D) [ ]
4. (A) [ ] (B) [ ] (C) [X] (D) [ ]
5. (A) [X] (B) [ ] (C) [ ] (D) [ ]
6. (A) [ ] (B) [X] (C) [ ] (D) [ ]
7. (A) [X] (B) [ ] (C) [ ] (D) [ ]
8. (A) [X] (B) [ ] (C) [ ] (D) [ ]
9. Per prevenire SQL Injection è necessario utilizzare *prepared statement* con *query parametriche* oppure *stored procedures*. Nel caso sia inevitabile utilizzare concatenazione di stringhe con valori inseriti da utenti per costruire statement o query SQL da eseguire, è necessario sanitizzare gli input provenienti dagli utenti per assicurare che non contengano comandi SQL dannosi (e.g.: filtrando caratteri o keyword non contentiti).
10. Il processo di *minification* riduce sensibilmente la dimensione dei file JavaScript, mantenendone inalterata la funzionalità. Tipiche operazioni effettuate in questo processo includono l'eliminazione dei whitespace e dei commenti. La riduzione della dimensione dei file si traduce in riduzione dei tempi del primo caricamento delle pagine web in cui i file sono inclusi, e riduzione del volume di traffico di rete.
11. Nell'ambito del Testing Web End-to-End, la fragilità si riferisce al fatto che il codice per l'esecuzione di test automatici tende a smettere di funzionare in presenza di cambiamenti minori nel layout oppure nell'aspetto delle pagine web dell'applicazione web sotto test. Questo tipicamente avviene perché, in seguito ai cambiamenti evolutivi dell'applicazione web, i selettori utilizzati dal test automatico non riescono più ad identificare correttamente gli elementi con cui interagire.