Here is a deeper review of your files:

reservationServices.ts
Strengths:

Good use of TypeScript types for Slot and CheckConflictArgs.
Helper functions are well-named and focused.
Comments and author/date info are present.
Issues & Suggestions:

Date Handling in generateWeeklySlots:

The function expects timeStart and timeEnd as strings, but directly passes them to new Date(). If these are not ISO strings, this may cause bugs or timezone issues.
Consider explicitly parsing date and time, e.g.:
The slot's date is derived from currentStart, but timeStart and timeEnd are stored as full ISO strings. This may be inconsistent with your type definition.
Unused/Commented Code:

The toTimestamp function is commented out. Remove if not needed.
Error Handling:

No try/catch or error handling for database operations. Consider wrapping Prisma calls in try/catch blocks.
Prisma Client Management:

The Prisma client is instantiated at the top level. For larger apps, consider managing its lifecycle more carefully (singleton pattern or dependency injection).
index.ts
Strengths:

Demonstrates CRUD operations with Prisma.
Uses async/await properly.
Disconnects Prisma client after main execution.
Issues & Suggestions:

Hardcoded Example Data:

The reservation data uses placeholder strings ("something something something"). Replace with actual timestamps or valid data for production.
Error Handling:

No error handling in main(). Add a catch block to handle exceptions:
Prisma Client Lifecycle:

Disconnect is handled in .then(), but if an error occurs, Prisma may not disconnect. See above for improved error handling.
Type Safety:

If possible, use TypeScript interfaces for reservation objects to ensure type safety.

License:

License is specified in the comment, but not in the repository. Consider adding a LICENSE file.
General Recommendations:

Add error handling to all async operations.
Validate and sanitize all input data before using in database queries.
Consider using environment variables for configuration (e.g., database connection).
Add unit tests for core logic, especially conflict checking and slot generation.
If you want specific code improvements or refactoring, let me know!