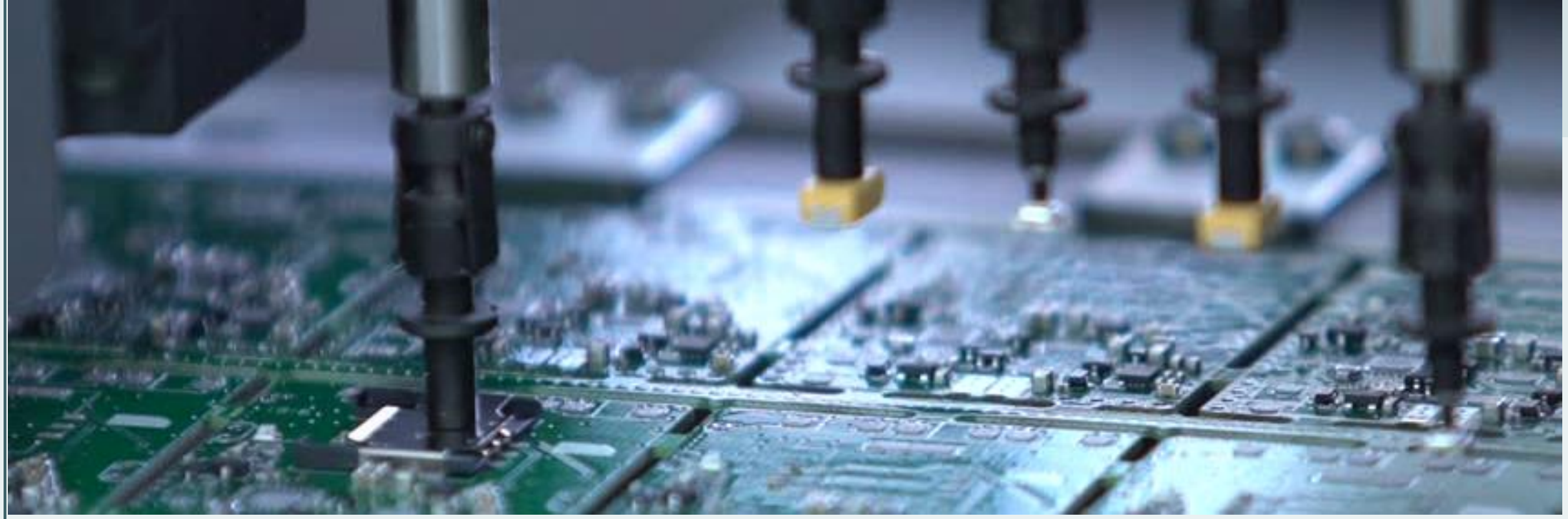
# Milestone 3: FaithQuestapi Development

AALaysha Gibson
cst391

# Project overview

- This milestone focused on:

- Building a REST API using Express and Node.js
- Writing the project in TypeScript
- Connecting the API to a MySQL relational database
- Implementing full CRUD functionality
- Testing endpoints using Postman



# Purpose of FaithQuest API

FaithQuest allows users to:
- Create daily scripture reflections
- Store notes and spiritual insights
- Mark entries as favorites
- Update existing reflections
- Delete journal entries

# Technology Stack

- Node.js – Server-side runtime environment
- Express – Framework used for routing and middleware
- TypeScript – Provides strong typing and improved code reliability
- MySQL – Relational database used to store journal entries
- Postman – Used to test and validate API endpoints

# REST API Design

The FaithQuest API follows REST conventions:

- Plural resource naming (/entries)
- Hierarchical routing
- HTTP verbs represent actions

CRUD Operations Implemented:

GET – Retrieve entries

POST – Create a new entry

PUT – Update an existing entry

DELETE – Remove an entry

# Example Endpoints

GET /devotionals
GET /devotionals/:id
POST /devotionals
PUT /devotionals/:id
DELETE /devotionals/:id


```
{

  "title": "Daily Strength",

  "scripture": "Philippians 4:13",

  "notes": "God gives me strength to overcome challenges.",

  "dateCreated": "2026-02-15",

  "isFavorite": true

}
```

# Database Integration &Challenges Encountered

Implementation steps included:

- Installing MySQL and required drivers
- Creating database schema
- Configuring environment variables
- Writing SQL queries for CRUD operations
- Handling database errors

- MySQL connection configuration issues
- Environment variables not loading properly
- TypeScript type mismatches
- Import/export syntax errors
- Git push conflicts
- Incorrect folder structure and package.json placement

# Pending Bugs or Issues

- No authentication or user accounts
  - Limited input validation
  - Basic error handling responses
  - No automated testing framework
  - No production deployment setup

# Lessons Learned

I have strengthened my understanding of:

- Designing RESTful APIs before coding
- Structuring backend applications properly
- Connecting Express to a relational database
- Using TypeScript for safer development
- Testing APIs consistently with Postman
- Managing version control using Git

# Conclusion

the Faith Quest REST API was successfully developed using Express, TypeScript, and a MySQL relational database. The application supports full CRUD functionality and follows RESTful design principles. Environment variables were implemented to securely manage database configuration settings, and Postman was used to test each endpoint. The API returns appropriate HTTP status codes and demonstrates proper separation of concerns through a layered architecture consisting of routes, controllers, services, and database modules. Overall, the system meets all project requirements and provides a scalable foundation for future enhancements.