Part 2 – Technology Requirements

A) Technology Requirements

Technology Use	Purpose	Pros	Cons
Frontend: JavaScript (React/Vite) + Tailwind CSS	Build a modern, responsive UI supporting interactive searching, filtering, maps, reviews, and booking pages.	- Fast development - Performance optimized - Rapid styling - Reusable utility classes - Easy integration	- Tailwind can reduce readability - Not ideal for very large, high custom designs - Requires build size optimization
Backend: Java Spring Boot (modular microservice-ready)	Provides REST APIs, business logic, security, booking workflow, verification process.	- High reliability - Strong security implementations - Large ecosystem & tooling - Excellent performance in production	- Higher memory usage compared to lightweight frameworks - Slower cold startup - Some boilerplate despite annotations
Database: MySQL on AWS RDS	Persist shops, users, services, bookings, reviews, verification logs	- Robust relational structure - Works naturally with Spring Boot + JPA - Highly optimized for read-heavy systems	- Requires careful indexing - Complex joins may slow down large datasets
Cache: Redis	Speed up search results, shop lookups, and rating calculations	- Extremely fast in-memory store - Ideal for high-read endpoints - Significantly reduces SQL load	- Requires proper cache invalidation to avoid stale data
CI/CD: Jenkins pipeline	Automates the software delivery process by continuously building, testing, and deploying the application	- Highly customizable - Large plugin ecosystem - Enables full DevOps lifecycle automation	- Maintenance overhead - Plugins may introduce dependency issues
Containers: Docker (multi-service setup)	Package the full system for easy deployment across all environments	- Consistent runtime environments - Lightweight compared to VMs - Easy to scale, distribute, or deploy	- Misconfiguration can accidentally expose ports or secrets

External APIs: Google Maps & Places API for shop info and location	Handle mapping, geolocation, address validation, shop details	- Offers many pre-built features via API. - Rich with data (coordinates, hours, photos)	- Data quality depends on what shop owners submit to Google - Rate-limited usage
Postman (API Testing)	Manual API testing & debugging during development	- Quickly tests endpoints without writing code - Supports all HTTP methods	- Requires external tools for automation - Limited for load testing

A) **Learning Plan**

Team Member	Role	Responsibilities	Current Skill Level	Skills To Develop	Learning Plan
Hossein Khanzadeh	UI Design	Will develop overall UI through React and Vite, using core components such as a search interface, interactive maps, shop profiles, and booking flow.	**Strength:** React, UI Design. **Needs Improvement:** Tailwind CSS(New), Figma.	**React:** - React components, props, and state. - React hooks. - Controlled input - React router - API Data fetching. - External library integration - State Management and Vite project structure **Tailwind CSS:** - Utility classes - Responsive Design - Component Styling - Reusable styles with @apply - Theme customization **Figma:** - Creating clean wireframes and mockups - Building page flows - Designing consistent typography / spacing - Interactive prototypes - working with components & variants **UI - Backend Integration:**	**November:** Learn Figma basics: frames, layout, typography, grids (Figma components guide). Build initial WrenchIT wireframes and flow diagrams. Refresh core React concepts; practice small components. **December:** Learn Tailwind utility classes, breakpoints, spacing, and dark mode. Create static screens using Tailwind from mockups. **January:** Implement real screens in React. Connect frontend to backend endpoints. Handle loading, error handling, empty states. Understand routing and param passing. Understand more about React Router. **February:** Learn and Integrate Google Maps

				- Calling Spring Boot APIs - Handling loading states, errors, empty states - Parsing JSON responses into UI components - Passing parameters in routes - Displaying dynamic results	JavaScript API. **<u>March</u>:** Learn about polishing UI components for consistency and visual bug resolution. Research on UI testing suites and best practices.
Tyson Ward-Dicks	Backend Developer	Handle data processing, workflows, user actions, and maintain API	-Debugging - Code quality and clean practice - Small-scale system integration - decent knowledge in containerized development, API implementation basics	- Better understanding of CI/CD pipeline - Improve microservice structural knowledge - Improve backend development knowledge	**<u>December</u>:** - CI/CD Pipelines (Version control workflows, cloud platforms, ect) **<u>January</u>:** - Microservices Architecture (microservice patterns, service communication, data management, ect) **<u>February</u>:** - API and server development (Design REST API, Authentication and authorization, validation and user input, caching strategies, ect) **<u>March</u>:** - Testing and Quality (Unit testing and integration testing, Test-driven development)
Hamzah Hafez	Back end Developer	- Build and maintain backend features, REST endpoints, and	- Comfortable with JavaScript and some AWS basics	- Spring Boot API development (controllers,	**<u>November</u>:** - Learn Spring Boot fundamentals and

		core workflows - Assist with cloud deployment and environment configuration - Support API integration with the frontend - Help debug, test, and improve backend reliability	 - Beginner in Spring Boot, microservices, and backend architecture - Needs deeper knowledge in DevOps and database integration	services, repositories) - Microservice structure and communication patterns - Efficient database handling with MySQL	build simple REST APIs - Practice JPA basics and understand project service boundaries **<u>December</u>:** - Implement backend features - Learn exception handling, validation, and basic microservice patterns - Begin integrating backend with MySQL **<u>January</u>:** - Learn AWS deployment workflow - Connect backend to AWS RDS - Work with Docker and Jenkins pipeline setup **<u>February</u>:** - Integrate backend endpoints with frontend - Improve API performance and start writing tests **<u>March</u>:** - Finalize backend features - Improve error handling, logging, and system stability - Assist with final AWS deployment

					steps
Fredrich Tan	Database Developer	- Design and maintain MySQL schema - Configuring Mysql on AWS RDS - Backend integration with Springboot + jpa - Data integrity, migrations and performance	**MySQL** - intermediate or needs a refresher **Spring Boot & Spring Data JPA** - Beginner **AWS RDS** - none **Performance optimization and migration** - Beginner	- Normalization -ERD Modelling -Indexing -Query optimization -Designing schema for invoices and reviews - Mapping -JPA repo -Fetch strategy -Handling of relationships -DTO mapping -create RDS instance -Create RDS instance -IAM roles & vpc -Secure DB connection -Parameter groups -AWS monitoring -Query profiling -caching (redis interaction) -Indexing decisions -schema refactoring -backup & migration	**<u>December</u>:** -Review relational modelling -Build draft WrenchIt ERD -Practice writing JOIN,FKs,indexes **<u>January</u>:** Learn entity mapping Build User/Shop/Review/ invoice models Connect Spring Boot **<u>February</u>:** -Deploy MYSQL on RDS -Set up VPC security Groups -Migrate schema to cloud **<u>March</u>:** -Add indexes -Test query -performance -Prepare migration scripts
Henrique Custodio	Lead Developer	Oversee system architecture across the frontend, backend, database, cloud, and CI/CD pipelines. Lead microservice design, integration, and communication	**Strong:** Spring Boot, Java, System architecture and Design, Docker, Cloud Platforms, DevOps, SQL, Linux administration. **Familiar:**	Improve microservice implementation (eg. Use of API gateway). Improve logging via centralized logging, tracing,	**<u>November</u>:** Research Backend Architecture and plan. Review key microservice patterns. Learn about Redis

		patterns. Implement core backend features/ microservices including authentication, booking flow, reviews, caching, and search. Set up and manage AWS infrastructure. Coordinate DevOps tasks such as Docker configuration, Jenkins automation, and environment setup.	Software architecture patterns, Microservice orchestration, Redis caching, React, API integration patterns, frontend–backend communication workflows, general JavaScript frontend development **Needs Review:** Google Maps API integration, React and Vite optimization, CSS & React Styling, advanced Jenkins automation.	metrics, monitoring, and reports. Improve testing practices with Load testing, Unit testing, and integration testing suites. Consistency with CI/CD and Jenkins. Production-grade caching and invalidation strategies. API performance tuning and distributed systems concepts.	caching and invalidation. **December:** Review AWS Infrastructure & DevOps. Review JWT auth, and other security practices. Review SQL schema design and implementations best practices. **January:** Research and configure Jenkins base CI/CD strategy. **February:** Resolve CORS issues, optimize builds, and assist with map/UI interactions. Refine Redis integration based on research for search/ratings. **March:** Research best testing practices and validate full application workflows.