2. Software Requirements Specification

2.1 Functional Requirements

User Requirements Specification

Total: 30 functions | Implementation: 20 functions

#	Function Description	Priority	Implemented
1	Register and log in via email	Must have	☑
2	Log in via Google	Should have	☑
3	Send money to a user by phone number	Must have	☑
4	View transaction history	Must have	☑
5	Check account balance	Must have	☑
6	Link a debit card securely	Must have	☑
7	Link a prepaid card	Must have	☑
8	Receive success/failure confirmation for transfers	Must have	☑
9	Edit profile and update phone number	Should have	☑
10	Reset password via email	Should have	☑
11	Block/unblock a linked card	Should have	☑
12	Report suspicious activity	Should have	✗
13	Scan QR code to pay another user	Should have	☑
14	View and download monthly account statements	Should have	☑
15	Add money to prepaid card from bank	Should have	✗
16	Set transfer limits per day/week	Could have	☑
17	Enable/disable notifications	Could have	✗
18	Access customer support chat	Could have	✗
19	Get in-app help/tutorials	Could have	☑
20	View app usage statistics (e.g. total amount sent)	Could have	✗
21	Add recipients to favorites for quick transfer	Should have	☑
22	Set a PIN for confirming transfers	Must have	☑
23	Schedule recurring transfers	Could have	✗
24	Export transactions as CSV/PDF	Could have	✗
25	Use biometric authentication (fingerprint/face)	Should have	✗
26	Multi-language support (English/Arabic toggle)	Won't have now	✗
27	Push notifications for incoming transfers	Should have	☑
28	Search/filter transactions by date or amount	Should have	☑
29	Request money from other users	Should have	✗
30	Create payment links to share externally	Could have	☑

System Requirements Specification

Total: 6 functions | Implementation: 4 functions

#	Function Description	Priority	Implemented
31	Process transfer requests in < 1 second	Must have	☑
32	Store passwords securely using hashing (e.g., bcrypt)	Must have	☑
33	Maintain a secure audit trail of all actions	Should have	☑
34	Auto-backup all data every 24 hours	Could have	✗
35	Support high concurrency without data loss	Must have	☑
36	Encrypt all sensitive data at rest and in transit	Must have	☑

Summary

- **36 total functions designed** (30 user + 6 system)
- **24 total functions selected for implementation** (marked ☑)
- Prioritized using **MoSCoW scheme**

2.2 Non-Functional Requirements

Categories Followed

- Performance
- Security
- Usability
- Maintainability
- Availability

Specification with Types

Requirement	Category
System must process 95% of transfers within 5 second	Performance
Passwords must be encrypted using bcrypt	Security
UI must be responsive and mobile-friendly	Usability
Code must follow PSR standards and be modular	Maintainability
System must have 99.9% uptime	Availability

Fit Criteria (Testable)

- **Performance**: Benchmark tests show 95% of API responses are < 5000ms.
- **Security**: Manual tests confirm no sensitive data is stored in plaintext.
- **Usability**: User feedback scores average 8+/10 in internal testing.
- **Maintainability**: 80%+ code coverage with unit tests.
- **Availability**: Monitored by uptime tools with 99.9% SLA simulation.

Impact on Architecture

- System will use a modular MVC architecture to ensure maintainability.
- APIs will be optimized and rate-limited to guarantee high performance.
- Security layer will include middleware for JWT validation and role checks.
- Frontend components will be decoupled to reduce load time and increase reusability.

2.3 Constraints

- **Development Timeline**: The project must be completed within the academic semester timeline.

- **Technology Constraints**: The system must be built using PHP for the backend and React for the frontend.
- **Database Constraints**: MySQL database must be used for data storage.
- **Platform Constraints**: The system must function correctly on major browsers (Chrome, Firefox, Safari, Edge).
- **Connectivity Constraints**: The system must handle intermittent connectivity gracefully.
- **Legal Constraints**: As an educational project, the system must operate with mock financial data only.

2.4 Validation Criteria

Approach to Requirements Validation

- **Traceability Matrix**: Each requirement is mapped to its corresponding test case
- **User Acceptance Testing**: Feedback from potential users evaluates if requirements meet expectations
- **Peer Reviews**: Team members review each other's work to ensure functionality matches requirements
- **Stakeholder Validation**: Regular meetings with instructors to validate requirements are being met

Success Criteria

- 95% of the "Must Have" requirements are successfully implemented and tested
- All implemented features pass their respective test cases
- The system demonstrates stability under normal usage conditions
- The application load time meets performance requirements
- Security measures are verified through penetration testing